

СОДЕРЖАНИЕ

Библиотека _____
„Кибернетического
Сборника“ _____

**ПРОБЛЕМЫ
МАТЕМАТИЧЕСКОЙ ЛОГИКИ
СЛОЖНОСТЬ АЛГОРИТМОВ
И КЛАССЫ ВЫЧИСЛИМЫХ ФУНКЦИЙ**

Сборник переводов

Под редакцией

В. А. КОЗМИДАДИ и А. А. МУЧНИКА

Сборник содержит работы по актуальным проблемам математической логики, еще не получившим достаточного освещения в отечественной литературе. Эти работы посвящены оценкам сложности алгоритмов и вычислений, классификациям рекурсивных функций и различным типам вычислительных устройств, связанных с такими классификациями. В частности, значительное место занимают исследования «ограниченных» машин Тьюринга и обобщений конечных автоматов. В ряде работ изучаются множества слов, распознаваемых обобщенными автоматами, причем обнаруживаются связи с грамматиками, введенными в работах Н. Хомского.

Книга рассчитана на лиц, интересующихся современными проблемами математической логики, теории алгоритмов, теории автоматов, математической лингвистики и теории вычислительных машин.

Редакция литературы по математическим наукам

Инд. 2-2-3

22-70

ПРОБЛЕМЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ

Редактор А. Г. Крылов

Художественный редактор В. И. Шаповалов

Технический редактор Л. П. Кондюкова

Корректор И. С. Додолева

Сдано в производство 5/XI 1969 г.

Подписано к печати 22/VII 1970 г.

Бумага тип. № 1

60×90¹/₁₆ = 13,50 бум. л.

Печ. л. 27. Уч.-изд. л. 24,53

Изд. № 1/5516 Цена 1 р. 94 к. Зак. 375

ИЗДАТЕЛЬСТВО «МИР» Москва, 1-й Рижский пер., 2

Ордена Трудового Красного Знамени
Ленинградская типография № 2 имени Евгении Соколовой
Главполиграфпрома Комитета по печати при Совете Министров СССР
Измайловский проспект, 29

ПРЕДИСЛОВИЕ

Теория алгоритмов возникла и оформилась в самостоятельную дисциплину в 30—40-х гг. нашего века. Примерно в это же время, а также в последующие годы были получены решения ряда классических алгоритмических проблем математической логики, алгебры и топологии. В дальнейшем наметилась связь теории алгоритмов с кибернетикой: проектированием вычислительных устройств, программированием, моделированием биологических объектов (хотя в довольно скромных пока масштабах), математической лингвистикой. Обнаружилось, что бывают алгоритмические проблемы («рекурсивно») разрешимые и неразрешимые. Последние даже различают по «степеням неразрешимости». Однако и теоретически и особенно практически важно уметь устанавливать, насколько «легко» разрешима та или иная проблема, какими средствами (типами алгоритмов, вычислительных устройств и т. п.) она разрешима, сколь сложными могут быть программы алгоритмов и соответствующие им вычисления.

В соответствии с этим стали изучаться ограниченные классы алгоритмов, реализуемых логическими сетями, контактными схемами, конечными и растущими автоматами. Много исследований, начиная с известных работ Дж. Риордана и К. Шеннона, было посвящено получению оценок сложности алгоритмов подобного рода, решающих ограниченные алгоритмические проблемы типа вычислений булевых функций, многочленов и т. д. Конечные автоматы и особенно их обобщения стали хорошим средством описания моделей реальных языков и языков программирования.

С другой стороны, изучение самих вычислимых функций привело к выделению некоторых довольно простых и вместе с тем достаточных для большинства приложений классов функций: примитивно рекурсивных, элементарных по Кальмару и т. д., а затем были предложены некоторые классификации рекурсивных функций в работах А. Гжегорчика, П. Акста, Дж. Клива, Р. В. Ричи. (Переводы этих работ вошли в предлагаемый читателю сборник, образуя в нем первый цикл, связанный с рекурсивными классификациями.)

Каждый алгоритм задается программой, согласно которой производятся операции над «текущей» информацией, находящейся в «памяти», а исходная информация определяется конкретной задачей, которую разрешает данный алгоритм. Указанные выше понятия уточняются при каждой формализации основного понятия — алгоритма. Последовательность операций при приме-

нении алгоритма $\mathcal{A}$ зависит от *исходных данных задачи* $x = (x_1, \dots, x_k)$ и называется *вычислением* $S_{\mathcal{A}}(x_1, \dots, x_k)$. Если сложность алгоритма, грубо говоря, характеризуется длиной его программы, то сложность вычисления характеризуется числом операций («временем»), количеством используемой «памяти» и другими величинами. Известны проблемы, разрешимые в принципе сравнительно простыми алгоритмами, основанными на переборе сверхастрономического числа вариантов (например, оптимальные алгоритмы в шахматах), но с очень «сложными» вычислениями, и, наоборот, встречаются довольно сложные алгоритмы, сравнительно быстро приводящие к цели — вычислению «функции» $\varphi_{\mathcal{A}}(x_1, \dots, x_k)$, значение которой является решением задачи с параметрами $x_1, \dots, x_k$. Всякое ограничение перебора связано с усложнением алгоритма. Особенность алгоритмов, реализуемых формулами, контактными схемами и логическими сетями без обратных связей, состоит в том, что число необходимых операций находится в прямой (а часто и в прямо пропорциональной) зависимости от «сложности» схемы, т. е. алгоритма. Однако такими схемами охватывается лишь очень узкий класс алгоритмов.

На оценках сложности алгоритмов основан алгоритмический подход к определению случайности и количества информации, развиваемый в последние годы в работах А. Н. Колмогорова ¹⁾, П. Мартин-Лёфа ²⁾, Л. И. Левина и др.

Вопросы сложности алгоритмов, вычисляющих булевы функции, рассматривались в работах В. А. Кузьмина ³⁾ и А. А. Маркова ⁴⁾. Оценкам сложности «начальных отрезков» неразрешимых алгоритмических проблем разрешения посвящены работы Я. М. Барздиня ⁵⁾, М. И. Кановича и Н. В. Петри ⁶⁾.

¹⁾ Колмогоров А. Н., Три подхода к определению понятия «количество информации», *Проблемы передачи информации*, 1, вып. 1 (1965), стр. 3—11.

²⁾ Мартин-Лёф П., О понятии случайной последовательности, *Теория вероятностей и ее применения*, 11, вып. 1 (1966), 198—200.

³⁾ Кузьмин В. А., Реализация функции алгебры логики автоматами, нормальными алгоритмами и машинами Тьюринга, сб. *Проблемы кибернетики*, вып. 13, 1965, стр. 75—96.

⁴⁾ Марков А. А., О нормальных алгорифмах, связанных с вычислением булевых функций, *Изв. АН СССР, серия матем.*, 31, вып. 1 (1967).

⁵⁾ Барздин Я. М., Сложность программ, *ДАН СССР*, 182, № 6 (1968), 1249—1252.

⁶⁾ Канович М. И., О сложности разрешающих алгорифмов, *ДАН СССР*, 186, № 5 (1969), 1008—1009; Канович М. И., Петри Н. В., Некоторые теоремы о сложности нормальных алгорифмов и вычислений, *ДАН СССР*, 184, № 6 (1969); Петри Н. В., Об алгорифмах, связанных с предикатами и булевыми функциями, *ДАН СССР*, 185, № 1 (1969); Петри Н. В., Сложность алгорифмов и время их работы, *ДАН СССР*, 186, № 1 (1969), 30—31.

Ряд работ посвящен другим вопросам сложности алгоритмов¹⁾.

С оценками сложности вычислений связан другой подход к классификации вычислимых функций. (Связь между аналитическим и «сложностным» подходами исследуется в статье Р. В. Ричи для «сложности», измеряемой длиной ленты машины Тьюринга, используемой при вычислении. В общей постановке этот вопрос рассмотрен в добавлении А. А. Мучника (стр. 123—138), в котором также дается обзор работ по классификации рекурсивных функций.)

Первые оценки сложности вычислений для конкретных предикатов и функций были получены в конце 50-х и начале 60-х гг. в работах Г. С. Цейтина, М. О. Рабина и Б. А. Трахтенброта и его учеников. Однако здесь сразу же пришлось столкнуться с большими трудностями, связанными с получением «хороших» нижних оценок (т. е. оценок, близких к верхним) для памяти и особенно времени вычисления. (Этим вопросам посвящены статьи Ф. Хенни, Дж. Хартманиса, П. М. Льюиса и Р. Стирнза, П. Фишера.) Специальный, но важный класс предикатов, вычислимых «в реальное время», рассматривался М. Рабином и А. Розенбергом (см. их статьи и в данном сборнике).

Развитие математической лингвистики и языков программирования привело к изучению специальных классов рекурсивных предикатов (на множествах слов) — контекстных и бесконтекстных (или контекстно-свободных) языков. Верхние оценки для сложности распознавания таких языков приводятся в статьях Дж. Хартманиса, П. М. Льюиса, Дж. Хартманиса и Р. Стирнза, Д. Янгера, Т. Касами и в добавлении к последней.

Абстрактный аксиоматический подход к изучению сложности алгоритмов и вычислений предложен М. Блюмом, который установил замечательные результаты о существовании «бесконечно оптимизируемых» в смысле сложности классов рекурсивных предикатов. Точный смысл высказанного утверждения и полное доказательство можно найти в двух статьях М. Блюма, заключающих этот сборник.

Мы надеемся, что настоящий сборник представит интерес для многих математиков, логиков, кибернетиков и тех, кто специализируется в области современного программирования и ма-

¹⁾ Брейбарт Ю. Я., Козмидиади В. А., О двух подклассах машин Тьюринга, сводимых к конечным автоматам, *ДАН СССР*, 187, № 1 (1969), 9—10; Остроухов Д. А., Об оценках сложности нормальных алгоритмов, *ДАН СССР*, 184, № 6 (1969), 1292—1293; Маранджян Г. Б., О некоторых свойствах асимптотически оптимальных рекурсивных функций, *Изв. АН Арм. ССР*, 4, вып. 1 (1969), 3—22; Маранджян Г. Б., Иерархия рекурсивных функций и асимптотическая аддитивность, *ДАН Арм. ССР*, 48, № 4 (1969), 193—197.

тематической лингвистики. По затронутым здесь проблемам на русском языке (кроме отдельных статей в научных журналах) опубликованы обзоры П. Фишера 1965 г.¹⁾ с обширной библиографией, А. Ахоу и Дж. Улмана²⁾ и Л. А. Гусева и И. М. Смирновой³⁾, а также курс лекций Б. А. Трахтенброта⁴⁾. В указанном курсе лекций подытожены и систематически излагаются результаты об оценках сложности вычислений на машинах Тьюринга (полученные вплоть до конца 1966 г.) и аксиоматическая теория сложности алгоритмов и вычислений, но почти не затрагиваются вопросы классификации (аналитической) рекурсивных функций. К сожалению, этот курс издан небольшим тиражом и мало доступен широкому кругу читателей. Это побудило нас включить в сборник переводы нескольких статей, обладающих большими методическими достоинствами (см., например, статью Ф. Хенни на стр. 223), содержание которых частично перекрывается работами советских математиков Я. М. Барздиня, Б. А. Трахтенброта, Р. В. Фрейвалда и других, опубликованными в советской печати (см. «Алгебра и логика» за 1963—1967 гг.).

Для понимания большинства статей достаточно знакомства с элементами теории алгоритмов, например, в объеме первых трех глав и § 12 книги А. И. Мальцева⁵⁾, а читателю статей о распознавании языков полезно было бы прочесть обзор Н. Хомского⁶⁾ и упомянутые выше обзоры по теории языков.

В. А. Козмидиади
А. А. Мучник

¹⁾ Фишер П., Многоленточные и бесконечные автоматы, Кибернетический сборник (новая серия), вып. 5, «Мир», 1968, стр. 64—80.

²⁾ Ахоу А., Улман Дж., Теория языков, Кибернетический сборник (новая серия), вып. 6, «Мир», 1969, стр. 145—183.

³⁾ Гусев Л. А., Смирнова И. М., Языки, грамматики и абстрактные автомные модели, I, II, *Автоматика и телемеханика*, № 4, № 5 (1968).

⁴⁾ Трахтенброт Б. А., Сложность алгоритмов и вычислений, Новосибирск, 1967.

⁵⁾ Мальцев А. И., Алгоритмы и рекурсивные функции, «Наука», М., 1965.

⁶⁾ Хомский Н., Формальные свойства грамматик, Кибернетический сборник (новая серия), вып. 2, «Мир», 1966, стр. 121—230.

НЕКОТОРЫЕ КЛАССЫ РЕКУРСИВНЫХ ФУНКЦИЙ ¹⁾

А. Гжегорчик

ВВЕДЕНИЕ

В этой статье исследуется возрастающая последовательность $\mathcal{E}^0, \mathcal{E}^1, \dots$ классов рекурсивных функций. Каждый класс $\mathcal{E}^n$ замкнут относительно операций подстановки и операции ограниченной рекурсии. Исходными являются примитивно рекурсивные функции. Поэтому $\mathcal{E}^n \subset \mathcal{R}$, где $\mathcal{R}$ — класс примитивно рекурсивных функций. Говоря точнее, $\mathcal{R} = \sum_n \mathcal{E}^n$. Следовательно, в определении класса $\mathcal{R}$ операцию рекурсии нельзя исключить или заменить операцией ограниченной рекурсии.

Отдельно рассматриваются классы $\mathcal{E}^0$ и $\mathcal{E}^3$. Для каждой функции $f \in \mathcal{E}^0$ существует число k_0 , такое, что $f(n) < n + k_0$ для всех n . Однако каждое рекурсивно перечислимое множество перечислимо некоторой функцией класса $\mathcal{E}^0$. Мы начнем с исследования класса $\mathcal{E}^3$. Это класс функций, элементарно вычисляемых (элементарных) по Кальмару.

1. ПРЕДВАРИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

1. Функции, нумерующие пары. Мы назовем *функциями, нумерующими пары*, каждую тройку функций P, Q, R , определенных над неотрицательными целыми числами и удовлетворяющих следующим условиям:

- (1) $P(Qz, Rz) = z$,
- (2) $Q(P(x, y)) = x$,
- (3) $R(P(x, y)) = y$.

Функции, которые удовлетворяют этим трем формулам, устанавливают взаимно однозначное соответствие между множеством пар неотрицательных целых чисел и всем множеством неотрицательных целых чисел. Функции Q и R являются обратными для функции P . Первый элемент пары, представимой z , является значением функции Qz , а второй элемент есть значение функции Rz .

¹⁾ Grzegorzczuk A., Some classes of recursive functions, *Rozprawy Matematyczne*, IV, Warszawa, 1953.

Например, следующие три функции являются функциями, нумерующими пары, с вышеописанными свойствами:

$$P(x, y) = 2^x \cdot (2y + 1) - 1;$$

Qz , равная наибольшему целому числу u , такому, что $z + 1$ делится на 2^u ;

$$Rz = \frac{\frac{z+1}{2^{Qz}} - 1}{2}.$$

Другими примерами функций, нумерующих пары, являются следующие функции I , K , L ¹⁾:

$$I(x, y) = \frac{1}{2}(x+y)(x+y+1) + x = \frac{(x+y)^2 + 3x + y}{2};$$

$Kz = z - \frac{1}{2}u(u+1)$, где u — наибольшее целое число, такое, что $\frac{1}{2}u(u+1) \leq z$;

Lz , равная неотрицательному решению уравнения $\frac{(y + Kz)^2 + 3Kz + y}{2} = z$, т. е. $Lz = uKz$.

Имеем $I(Kz, Lz) = z$, $K(I(x, y)) = x$, $L(I(x, y)) = y$.

Функции, нумерующие пары, позволяют нам образовать функции троек, например

$$T(x, y, z) = P(x, P(y, z)) \text{ или } S(x, y, z) = I(x, I(y, z)),$$

которые устанавливают соответствие между множеством троек чисел и самим множеством чисел. Обратными к T функциями являются

$$T_1u = Qu, \quad T_2u = QRu, \quad T_3u = RRu.$$

Аналогично все конечные наборы чисел могут быть представлены натуральными числами.

Буквы P , Q , R и I , K , L будут впоследствии обозначать тройки функций, нумерующих пары.

2. Универсальные функции. Все рассматриваемые в дальнейшем функции определены на множестве неотрицательных целых чисел и принимают целые значения. Мы будем употреблять названия «целое число» и «число» только в смысле «неотрицательное целое число». Прописными буквами $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$, $\mathcal{E}^n$ обозна-

¹⁾ Приведенные в оригинале функции I , K , L не удовлетворяют свойствам функций, нумерующих пары. Поэтому мы их заменили функциями, связанными с обычным пересчетом пар натуральных чисел вдоль прямых $x+y = \text{const.}$ — Прим. перев.

чаются классы функций. Если $\mathcal{X}$ — класс функций, то через $\mathcal{X}_n$ обозначается класс функций от n аргументов, принадлежащих $\mathcal{X}$.

Класс $\mathcal{X}$ **замкнут относительно операции подстановки**, если он замкнут относительно следующих трех операций.

1. Операция подстановки (супепозиции) функций. Если $\mathcal{X}$ включает в себя функции f и g , заданные выражениями

$$f(x_1, x_2, \dots, x_{k-1}, x_k, x_{k+1}, \dots, x_n), g(y_1, \dots, y_m),$$

то он включает в себя функцию, называемую **подстановкой функции g в функцию f** вместо k -го переменного, которая записывается следующим образом:

$$f(x_1, \dots, x_{k-1}, g(y_1, \dots, y_m), x_{k+1}, \dots, x_n).$$

2. Операция идентификации переменных. Если класс $\mathcal{X}$ включает в себя функцию $f(x_1, \dots, x_j, \dots, x_k, \dots, x_m)$, то этот класс включает в себя также функцию, полученную идентификацией переменных x_j и x_k и заменой их переменной y и записываемую как $f(x_1, \dots, y, \dots, y, \dots, x_m)$. Переменная y отличается от всех переменных x_i .

3. Операция подстановки константы. Если класс $\mathcal{X}$ включает в себя функцию $f(x_1, \dots, x_k, \dots, x_m)$, то он также включает в себя функцию $f(x_1, \dots, 0, \dots, x_m)$, получаемую из функции f подстановкой 0 вместо k -й переменной.

Например, класс $\mathcal{A}$ линейных функций, т. е. класс функций f вида

$$f(x_1, \dots, x_n) = k_1x_1 + k_2x_2 + \dots + k_nx_n + k_0, \text{ где } k_1, \dots, k_n > 0,$$

является наименьшим классом, содержащим исходные функции $x + 1$, $x + y$ и замкнутым относительно операций подстановки.

Функция $F(x_1, \dots, x_n, t)$ от $n + 1$ аргументов называется **универсальной функцией** для класса $\mathcal{X}_n$ при условии, что каждая функция $f(x_1, \dots, x_n)$ от n аргументов принадлежит $\mathcal{X}_n$ тогда и только тогда, когда существует число t , такое, что для всех $x_1, \dots, x_n$ $f(x_1, \dots, x_n) = F(x_1, \dots, x_n, t)$.

Например, функция $F(x, t) = (Qt + 1)x + Rt$ универсальна для класса $\mathcal{A}_1$, функция $F(x_1, x_2, t) = (T_1t + 1)x_1 + (T_2t + 1)x_2 + T_3t$ универсальна для класса $\mathcal{A}_2$.

Очевидно, что если класс $\mathcal{X}$ содержит парные функции и замкнут относительно операции подстановки, то универсальная функция $F_n(x_1, \dots, x_n, t)$ для класса $\mathcal{X}_n$ может быть определена с помощью универсальной функции $F(x, t)$ для класса $\mathcal{X}_1$. Именно, можно положить

$$F_2(x_1, x_2, t) = F(I(x_1, x_2), t).$$

И вообще

$$F_n(x_1, \dots, x_n, t) = F(I(x_1, I(x_2, \dots, I(x_{n-1}, x_n) \dots)), t).$$

Если класс функций $\mathcal{X}$ замкнут относительно операций подстановки и содержит функцию $x + 1$, то универсальная функция для класса $\mathcal{X}_n$ не принадлежит классу $\mathcal{X}$. Если $F(x, t)$ — универсальная функция для класса $\mathcal{X}_1$, то для F , принадлежащей классу $\mathcal{X}$, функция $F(x, x) + 1$ принадлежит $\mathcal{X}_1$. Следовательно, для некоторого t $F(x, x) + 1 = F(x, t)$. Это приводит к противоречию: $F(t, t) + 1 = F(t, t)$.

3. Отношения данного класса. Мы будем отождествлять множества с отношениями одного аргумента. Понятие отношения будет связано с понятием функции данного класса. Скажем, что отношение n элементов $R(x_1, \dots, x_n)$ является *отношением класса* $\mathcal{X}$ тогда и только тогда, когда существует такая функция f класса $\mathcal{X}$, что имеет место следующая эквивалентность:

$$R(x_1, \dots, x_n) \equiv f(x_1, \dots, x_n) = 0 \text{ для всех } x_1, \dots, x_n.$$

Пусть $x \dot{-} y$ обозначает разность, определенную для неотрицательных целых чисел: $x \dot{-} y = x - y$, если $x \geq y$, и $x \dot{-} y = 0$, если $x < y$.

Теорема 1.1. Если $\mathcal{X}$ является классом функций, замкнутым относительно операции подстановки, и $\mathcal{X}$ включает функции $x + 1$, $x + y$, $x \dot{-} y$, то множество отношений класса $\mathcal{X}$ замкнуто относительно операций исчисления высказываний.

Доказательство. Если $R(x_1, \dots, x_n)$ и $S(y_1, \dots, y_m)$ являются отношениями класса $\mathcal{X}$, то существуют функции f и g , которые удовлетворяют эквивалентностям

$$R(x_1, \dots, x_n) \equiv f(x_1, \dots, x_n) = 0, \quad S(y_1, \dots, y_m) \equiv g(y_1, \dots, y_m) = 0.$$

В соответствии со следующими очевидными свойствами арифметики неотрицательных целых чисел¹⁾:

$$a = 0, b = 0. \equiv a + b = 0, \quad \sim(a = 0) \equiv 1 \dot{-} a = 0$$

можно вывести эквивалентности

$$\sim R(x_1, \dots, x_n) \equiv 1 \dot{-} f(x_1, \dots, x_n) = 0,$$

$$R(x_1, \dots, x_n). S(x_1, \dots, x_n). \equiv f(x_1, \dots, x_n) + g(x_1, \dots, x_n) = 0.$$

Эти эквивалентности показывают, что отношение, являющееся дополнением к отношению R , и отношение, являющееся

¹⁾ $A.B$ здесь и далее обозначает конъюнкцию утверждений A и B ; $\sim A$ — отрицание A . — Прим. перев.

логическим произведением отношений R и S , принадлежат классу $\mathcal{K}$. Поскольку все остальные конечные логические операции (операции исчисления высказываний) могут быть определены через конъюнкцию (логическое произведение) и отрицание (дополнение), логические операции не выводят из класса $\mathcal{K}$.

Следствие 1.2. *Вместо функций $x + 1$, $x + y$ и $x \div y$ с целью определения логических операций исчисления высказываний на отношениях можно использовать функцию x^y .*

Доказательство. Имеем

$$\sim(a = 0) \equiv 0^a = 0, \quad a = 0.b = 0. \equiv .a^{0b} = 0.$$

Следовательно,

$$\sim R(x_1, \dots, x_n). \equiv .0^f(x_1, \dots, x_n) = 0,$$

$$R(x_1, \dots, x_n). S(y_1, \dots, y_m). \equiv .(f(x_1, \dots, x_n))^{0g(y_1, \dots, y_m)} = 0.$$

Пусть $\mathcal{P}$ будет класс полиномов, определенных следующим образом: $\mathcal{P}$ является наименьшим классом, который включает исходные функции $x + 1$, $x + y$, $x \div y$, $x \cdot y$ и замкнут относительно операций подстановки.

Например, отношения $<$, $\leq$, $=$, $\neq$ принадлежат классу $\mathcal{P}$, поскольку имеют место эквивалентности

$$x < y. \equiv .((x + 1) \div y) = 0,$$

$$x \leq y. \equiv .(x \div y) = 0,$$

$$x = y. \equiv .(x \div y) + (y \div x) = 0,$$

$$x \neq y. \equiv .1 \div [(x \div y) + (y \div x)] = 0.$$

2. ЭЛЕМЕНТАРНЫЕ ФУНКЦИИ

Определение. Понятие элементарной функции введено Кальмаром¹⁾. Этот класс шире, чем классы, рассмотренные в предыдущем разделе. В большинстве случаев для практических приложений оказывается достаточно элементарных функций. В соответствии со слегка упрощенным²⁾ определением Кальмара **класс элементарных функций $\mathcal{E}$** — это наименьший класс,

(1°) включающий исходные функции $x + 1$, $x + y$, $x \div y$,

(2°) замкнутый относительно следующих операций:
операции подстановки,

¹⁾ См. Петер [4], стр. 82, 83 русского перевода.

²⁾ Приводимое определение допускает дальнейшие упрощения. Именно, можно ограничиться лишь функциями $x + 1$, $x \div y$ и операциями подстановки и ограниченного умножения. — Прим. ред.

операции ограниченного суммирования и ограниченного умножения, которые устанавливают, что если $F(x_1, \dots, x_n, t)$ принадлежит классу $\mathcal{E}$, то этот класс включает в себя функции f и g , определенные так:

$$f(x_1, \dots, x_n, y) = \sum_{i \leq y} F(x_1, \dots, x_n, i),$$

$$g(x_1, \dots, x_n, y) = \prod_{i \leq y} F(x_1, \dots, x_n, i).$$

Здесь операции $\sum_{i \leq y}$ и $\prod_{i \leq y}$ обозначают конечную сумму и произведение ¹⁾

$$\sum_{i \leq y} F(u, i) = F(u, 0) + F(u, 1) + \dots + F(u, y),$$

$$\prod_{i \leq y} F(u, i) = F(u, 0) \cdot F(u, 1) \cdot \dots \cdot F(u, y).$$

Например, $x \cdot y$, x^y , $x!$ являются элементарными функциями:

$$x \cdot y = \sum_{i \leq y} ((x + i) \dot{-} i) \dot{-} x = ((x + 0) \dot{-} 0 + (x + 1) \dot{-} 1 + \dots \\ \dots + (x + y) \dot{-} y) \dot{-} x.$$

Положим

$$f(x, y) = x^{y+1} = \prod_{i \leq y} (x + i) \dot{-} i,$$

$$x^y = (1 \dot{-} (1 \dot{-} y)) \cdot f(x, y \dot{-} 1) + (1 \dot{-} y)$$

(что необходимо для $x^0 = 1$) и

$$g(x) = \prod_{i \leq x} (i + 1), \quad x! = g(x \dot{-} 1).$$

Другие элементарные функции будут определены посредством некоторых операций, которые в свою очередь могут быть определены с помощью операций, указанных в определении класса $\mathcal{E}$.

Операция ограниченного минимума. Это операция перехода от функции $F(u, x)$ от $n + 1$ аргументов к функции $f(u, y) = \mu x \leq y [F(u, x) = 0]$ от $n + 1$ аргументов, определенной следующим образом:

$$\mu x \leq y [F(u, x) = 0] = \begin{cases} \text{наименьшему } x \leq y, \text{ такому, что } F(u, x) = \\ = 0, \text{ если такой } x \text{ существует,} \\ 0, \text{ если не существует } x \leq y, \text{ для кото-} \\ \text{рого } F(u, x) = 0. \end{cases}$$

¹⁾ Для упрощения обозначений мы употребляем готические буквы при обозначении наборов переменных, например u есть сокращение, обозначающее $u_1, \dots, u_n$.

Теорема 2.1. *Класс $\mathcal{E}$ замкнут относительно операции ограниченного минимума.*

Доказательство. Если $F(u, y)$ принадлежит классу $\mathcal{E}$ и если функция g удовлетворяет условию

$$g(u, x) = \mu y \leq x [F(u, y) = 0],$$

то функция g также принадлежит классу $\mathcal{E}$. А именно, функция g может быть определена посредством операции ограниченного суммирования следующим образом. Сначала определим вспомогательную функцию f

$$(i) \quad f(u, y) = \sum_{i \leq y} (1 \dot{-} F(u, i)),$$

а затем положим

$$(ii) \quad g(u, x) = \left(\sum_{y \leq x} (1 \dot{-} f(u, y)) \right) \cdot \left(1 \dot{-} \left(\sum_{y \leq x} (1 \dot{-} f(u, y)) \right) \dot{-} x \right).$$

Заметим в первую очередь, что $f(u, y) = 0$ в том и только в том случае, если для каждого $i \leq y$ мы имеем $F(u, i) \neq 0$. Условия $f(u, y) = 0$ и $1 \dot{-} f(u, y) = 1$ эквивалентны.

Следовательно, сумма $\sum_{y \leq x} (1 \dot{-} f(u, y))$ содержит столько единиц, сколько найдется чисел среди $y = 0, 1, 2, \dots, x$, для которых из $i \leq y$ следует, что $F(u, i) \neq 0$. Если среди чисел $y = 0, 1, 2, \dots, x$ существует такое наименьшее число $a \leq x$, что $F(u, a) = 0$, тогда, конечно, только для чисел $y < a$ верно, что для каждого $i \leq y$ имеем $F(u, i) \neq 0$. Следовательно, только для $y < a$ верно, что $1 \dot{-} f(u, y) = 1$; отсюда

$$\sum_{y \leq x} (1 \dot{-} f(u, y)) = a,$$

потому что эта сумма содержит столько единиц, сколько существует чисел меньших, чем a . Поскольку $a \leq x$, эта сумма меньше, чем $x + 1$. Отсюда следует, что второй сомножитель в определении $g(u, x)$ в этом случае равен 1 и $g(u, x) = a$.

Если такого числа $a \leq x$, что $F(u, a) = 0$, не существует, то для каждого $y \leq x$ верно, что для всякого $i \leq y$ $F(u, i) \neq 0$, т. е. для всякого $y \leq x$ $1 \dot{-} f(u, y) = 1$ и

$$\sum_{y \leq x} (1 \dot{-} f(u, y)) = x + 1,$$

ибо эта сумма содержит $x + 1$ единицу. Второй сомножитель в определении функции g в таком случае равен нулю и $g(u, x) = 0$, что также согласуется со значением операции ограниченного минимума, как она была определена выше.

Заметим, что в (i) и (ii) операция минимума была определена посредством операции, которая преобразует функцию $F(u, x)$ в функцию $f(u, y)$, определенную следующим образом:

$$f(u, y) = \sum_{i \leq y} 1 \dot{-} F(u, i).$$

Назовем эту операцию **операцией суженного ограниченного суммирования**. Это замечание может быть выражено в следующей форме общего заключения.

Следствие 2.1а. Если класс $\mathcal{X}$ замкнут относительно операций подстановки и суженного ограниченного суммирования и содержит функции $x \dot{-} y$ и $\sigma(x, y) = x \cdot (1 \dot{-} y)$, то $\mathcal{X}$ замкнут относительно операции ограниченного минимума.

Элементарные отношения. Отношения класса $\mathcal{E}$ будем называть **элементарными**.

Будем говорить, что отношение $T(u, y)$ определено из отношения R с помощью операции навешивания ограниченного квантора существования, если имеет место следующая эквивалентность:

$$T(u, y) \equiv \sum_{x \leq y} R(u, x) \equiv \sum_x \{x \leq y \cdot R(u, x)\}^1.$$

Отношение $S(u, y)$ определено посредством операции навешивания ограниченного квантора общности, если верна эквивалентность

$$S(u, y) \equiv \prod_{x \leq y} R(u, x) \equiv \prod_x \{x \leq y \rightarrow R(u, x)\}^2.$$

Теорема 2.2а. Класс элементарных отношений замкнут относительно логических операций навешивания ограниченных кванторов.

Доказательство. Если R является элементарным отношением, т. е. существует такая элементарная функция f , что

$$R(u, x) \equiv f(u, x) = 0,$$

а отношения T и S определены, как это было сделано выше, то существуют элементарные функции g и h , такие, что

$$(\alpha) \quad T(u, y) \equiv h(u, y) = 0,$$

$$(\beta) \quad S(u, y) \equiv g(u, y) = 0.$$

¹⁾ $\sum_{x \leq y} A(x)$ означает: «существует $x \leq y$, такой, что $A(x)$ ». — Прим.

перев.
²⁾ $\prod_{x \leq y} A(x)$ означает: «для всех $x \leq y$ имеет место $A(x)$ ». — Прим.
перев.

Такие функции могут быть прямо определены с помощью операций суммирования и умножения:

$$h(u, y) = \prod_{x \leq y} f(u, x), \quad g(u, y) = \sum_{x \leq y} f(u, x).$$

Легко доказать, что обе эти функции удовлетворяют эквивалентностям (α) и (β) .

Можно также доказать следующее утверждение.

Теорема 2.2b. Если класс $\mathcal{X}$ замкнут относительно операции ограниченного минимума и подстановки, то класс отношений класса $\mathcal{X}$ замкнут относительно операции навешивания ограниченного квантора существования.

Доказательство. Если $f \in \mathcal{X}$ и $f(u, x) = 0 \equiv R(u, x)$, то функция

$$h(u, y) = f(u, \mu x \leq y [f(u, x) = 0])$$

принадлежит классу $\mathcal{X}$.

Легко видеть, что

$$h(u, y) = 0. \equiv . \sum_{x \leq y} f(u, x) = 0.$$

Следовательно, класс $\mathcal{X}$ также замкнут относительно операции навешивания ограниченного квантора существования.

Также можно доказать, что если класс $\mathcal{X}$ замкнут относительно операций исчисления высказываний, то класс $\mathcal{X}$ замкнут относительно операции навешивания ограниченного квантора общности. Это следует из хорошо известного логического правила де Моргана

$$\prod_{x \leq y} R(u, x). \equiv . \sim \left(\sum_{x \leq y} \sim R(u, x) \right).$$

Из теоремы 1.1 следует, что класс элементарных отношений замкнут относительно операций исчисления высказываний. Так, отношения «меньше чем», «равно», «делит» и класс¹⁾ простых чисел являются элементарными, поскольку они определяются следующим образом:

$$x \leq y \equiv x \dot{-} y = 0; \quad x = y. \equiv . x \leq y. y \leq x;$$

$$x < y \equiv x \leq y. \sim (x = y); \quad x | y. \equiv \sum_{z \leq y} x \cdot z = y;$$

$$\text{prime}(x) \equiv \left\{ \prod_{y \leq x} y | x: \rightarrow : y = 1. \vee . y = x \right\}. x \geq 2.$$

¹⁾ Автор называет некоторый класс (чисел) элементарным в том и только том случае, когда отношение «быть элементом класса» является элементарным. — Прим. ред.

Отсюда, например, следует, что указанные ниже функции являются элементарными:

$$\left[\frac{x}{y} \right] = \mu z \leq x [x + 1 \leq y \cdot (z + 1)]$$

(целая часть частного x/y);

$$r(x, y) = x \dot{-} \left(y \cdot \left[\frac{x}{y} \right] \right)$$

(остаток от деления x на y);

$$x^+ = x^x; \quad x^{++} = x^{x^+}$$

(вспомогательные функции);

$$\left[\sqrt[y]{x} \right] = \mu z \leq x \left[z^y \leq x \cdot \left\{ \prod_{t \leq x} t^y \leq x \rightarrow t \leq z \right\} \right]$$

(целая часть корня).

Операции максимума. В дальнейшем мы используем две следующие операции максимума:

$$(\max x \leq z) [R(u, x)] = \mu x \leq z \left[R(u, x) \cdot \left\{ \prod_{t \leq z} R(u, t) \rightarrow t \leq x \right\} \right]$$

(наибольшее $x \leq z$, для которого имеет место отношение $R(u, x)$) и

$$\text{Max } F(u, x) \text{ (для } x \leq z) = F(u, \mu x \leq z \left\{ \prod_{t \leq z} F(u, t) \leq F(u, x) \right\})$$

(максимальное значение функции $F(u, x)$ для $x \leq z$). Как следует из их определений, эти операции не выводят из класса $\mathcal{E}$.

Простые числа. Простые числа будут играть важную роль в дальнейших рассмотрениях. Последовательность простых чисел определена как элементарная. Следующие функции и отношения являются элементарными:

$$\exp(y, k) = (\max x \leq y) (k^x | y)$$

(наибольший показатель x , для которого y еще делится на k^x);

$$xNy. \equiv .\text{prime}(x). \text{prime}(y). x > y. \prod_{z \leq x} \{y < z. \text{prime } z. \rightarrow z = x\}$$

(x является следующим простым числом за y);

$$\begin{aligned} \text{(I) } x \text{ Pr } k. &\equiv . \sum_{v \leq (k+2)^{++}} \left\{ \exp(v, 2) = 2. \text{prime}(x). \exp(v, x) = \right. \\ &= k + 2. \prod_{t \leq (k+2)^+} \prod_{z < t} \left[\text{prime}(z). \text{prime}(t). z | v. t | v. \rightarrow \right. \\ &\quad \left. \left. \rightarrow . \sum_{w \leq t} \{w | v. wNz. \exp(v, w) = \exp(v, z) + 1\} \right] \right\} \end{aligned}$$

(x является k -м простым числом);

$$(II) \quad p_k = \mu x \leq (k+2)^+ [x \text{ Pr } k]$$

(k -е простое число).

Согласно последнему определению, $2 = p_0$, $3 = p_1$ и т. д. Число v , которое фигурирует в определении (I), есть число вида

$$v = 2^2 \cdot 3^3 \cdot 5^4 \dots p_k^{k+2}.$$

В соответствии с этим определением это число удовлетворяет тому условию, что в его разложении на простые сомножители показатель каждого следующего простого числа на единицу больше, чем показатель предыдущего простого числа. Поскольку имеет место известное из теории чисел неравенство

$$p_k \leq (k+2)^{k+2} = (k+2)^+,$$

то (II) является адекватным определением последовательности p_k .

Представление конечных последовательностей чисел. Определенные выше функции позволяют многими способами представлять конечные последовательности чисел одним числом. Этот метод часто применяется в теории вычислимых функций. Конечную последовательность чисел $m_0, \dots, m_k$ представляет число $m = 2^{m_0} \cdot 3^{m_1} \dots p_k^{m_k}$.

Числа $m_0, \dots, m_k$ являются показателями степеней соответствующих простых чисел в разложении числа m : $m_i = \exp(m, p_i)$. Поэтому вместо выражения:

Существует конечная последовательность чисел $m_0, \dots, m_k$, таких, что

$$R(m_0, \dots, m_k),$$

можно сказать:

Существует такое неотрицательное число m , что

$$R(\exp(m, p_0), \dots, \exp(m, p_k)).$$

Это представление конечных последовательностей употребляется прежде всего для элементарных формулировок рекурсивных определений¹⁾.

¹⁾ Другой метод представления конечных последовательностей был дан Гёделем. Для каждой конечной последовательности чисел $a_0, \dots, a_n$ существуют такие два числа u, v , что $r(u, 1 + v(z+1)) = a_z$ (см. Робинсон [5] и Гёдель [1]).

Операция ограниченной рекурсии. Класс $\mathcal{X}$ замкнут относительно операции ограниченной рекурсии, если он удовлетворяет следующим условиям:

если g, h, j — функции класса $\mathcal{X}$ и функция f удовлетворяет условиям

$$\begin{aligned} & (a) \quad f(u, 0) = g(u), \\ (I) \quad & (b) \quad f(u, x+1) = h(u, x, f(u, x)), \\ & (c) \quad f(u, x) \leq j(u, x), \end{aligned}$$

то функция f также принадлежит классу $\mathcal{X}$.

Условия (a) и (b) — обычные условия рекурсивных определений функций, применяемых в арифметике.

Условие (a) определяет начальное значение f , а условие (b) выражает значение f для следующего числа в зависимости от ее значения для предыдущего числа с помощью функции h . При этом значение f определяется для каждого значения переменного x . Таким образом, функция $f(u, x)$ однозначно определяется условиями (a) и (b). Условие (c) ограничивает эту возможность определения теми функциями, которые не превосходят некоторую другую функцию, содержащуюся в классе $\mathcal{X}$.

Примеры. Если класс $\mathcal{X}$ замкнут относительно операции ограниченной рекурсии¹⁾ и содержит функции $S(x) = x + 1$ и x^y , то он также включает функции $x + y$ и $x \cdot y$, поскольку эти функции могут быть определены посредством ограниченной рекурсии. Полагая

$$\begin{aligned} U_2(x, y) &= y^{x^0} = y, \\ g_1(y) &= U_2(y, y), \quad h_1(y, x, z) = U_2(y, U_2(x, S(z))), \\ j_1(y, x) &= S(y)^{S(x)}, \end{aligned}$$

можно легко доказать, что функция $f(y, x) = y + x$ удовлетворяет схеме

$$\begin{aligned} & (a) \quad f(y, 0) = g_1(y), \\ (A) \quad & (b) \quad f(y, x+1) = h_1(y, x, f(y, x)), \\ & (c) \quad f(y, x) \leq j_1(y, x). \end{aligned}$$

Имея функцию $y + x$, аналогичным образом положим

$$g_2(y) = U_2(y, 0), \quad h_2(y, x, z) = U_2(x, y + z)$$

и видим, что умножение удовлетворяет условиям

$$\begin{aligned} & (a) \quad y \cdot 0 = g_2(y), \\ (B) \quad & (b) \quad y \cdot (x+1) = h_2(y, x, y \cdot x), \\ & (c) \quad y \cdot x \leq j_1(y, x). \end{aligned}$$

¹⁾ И подстановки. — Прим. ред.

В дальнейшем, применяя операцию рекурсии, мы не будем ни записывать тождественные функции U_1, U_2 ($U_1(x, y) = x = U_2(y, x)$), которые нужны были лишь из формальных соображений, ни определять вспомогательные функции g, h , а будем прямо записывать формулы рекурсии, например, заменив формулы в (А) и (В) следующими более простыми эквивалентными им формулами:

$$\begin{aligned} y + 0 &= y, & y \cdot 0 &= 0, \\ y + (x + 1) &= S(y + x), & y \cdot (x + 1) &= y + (y \cdot x), \\ y + x &\leq Sx^{Sy}; & y \cdot x &\leq Sx^{Sy}. \end{aligned}$$

Эти формулы, конечно, имеют тот же смысл, что и формулы в (А) и (В). Далее, если мы определяем функцию одного аргумента $f(x)$, то схема (I) легко приводится к схеме

$$\begin{aligned} &(a) \quad f(0) = g(0), \\ (II) \quad &(b) \quad f(x + 1) = h(x, f(x)), \\ &(c) \quad f(x) \leq j(x). \end{aligned}$$

Определение функции $x!$ может служить примером применения схемы (II):

$$\begin{aligned} 0! &= S(0), \\ (x + 1)! &= x! \cdot S(x), \\ x! &\leq x^x. \end{aligned}$$

Теорема 2.3. *Класс $\mathcal{E}$ замкнут относительно операции ограниченной рекурсии.*

Доказательство. Если функции f, g, h, j удовлетворяют условиям (a), (b) и (c) схемы (I), то

(1) $y = f(u, x)$ тогда и только тогда, когда существует конечная последовательность чисел $m_0, \dots, m_x$, такая, что

$$\begin{aligned} m_0 &= g(u), \quad m_1 = h(u, 0, m_0), \quad m_2 = h(u, 1, m_1), \dots, \\ m_x &= h(u, x - 1, m_{x-1}), \quad y = m_x. \end{aligned}$$

Таким образом, числа $m_0, \dots, m_x$ удовлетворяют условиям $m_0 = g(u)$, $m_{k+1} = h(u, k, m_k)$, которые точно определяют каждое из них. В соответствии с описанным выше методом представления конечных последовательностей числам $m_0, \dots, m_x$ можно сопоставить число $m = 2^{m_0} \cdot 3^{m_1} \cdot \dots \cdot p_x^{m_x}$. Таким образом, $m_i = \exp(m, p_i)$; следовательно, индуктивное условие $m_{k+1} = h(u, k, m_k)$ эквивалентно следующему:

$$\exp(m, p_{k+1}) = h(u, k, \exp(m, p_k)).$$

Следующая эквивалентность вытекает из эквивалентности (1). Полагая

$$F(x, u) = p_x^{(x+2) \cdot \text{Max } j(u, z)} \text{ при } z \leq x,$$

имеем

$$(2) \quad y = f(u, x) \equiv \sum_{m \leq F(x, u)} \left\{ \exp(m, 2) = g(u) \cdot \wedge \cdot y = \right. \\ \left. = \exp(m, p_x) \cdot \prod_{k < x} [\exp(m, p_{k+1}) = h(u, k, \exp(m, p_k))] \right\}.$$

Поскольку p_x является наибольшим простым сомножителем в разложении m и $m_i \leq j(u, i)$, то $\exp(m, p_i) \leq \text{Max } j(u, i)$ для $i \leq x$. Таким образом, верхняя оценка для числа m достаточна. Обозначим через $R(y, u, x)$ отношение в правой части эквивалентности (2). Отношение R является элементарным, если элементарны функции g, h, j . Из эквивалентности (2) следует, что функция f может быть определена через операцию ограниченного минимума посредством отношения R так:

$$f(u, x) = \mu y \leq j(u, x) [R(y, u, x)].$$

Итак, функция f элементарна.

Эквивалентные определения класса элементарных функций ¹⁾.

Пусть $\mathcal{E}'$ — наименьший класс функций,

1° содержащий исходные функции $x+1$, $x \div y$, x^y и

2° замкнутый относительно следующих операций:

операции подстановки,

операции ограниченного минимума.

Теорема 2.4. *Класс $\mathcal{E}'$ совпадает с классом $\mathcal{E}$.*

Доказательство. Первое включение $\mathcal{E}' \subset \mathcal{E}$ следует из того, что x^y принадлежит классу $\mathcal{E}$ и что этот класс замкнут относительно операции ограниченного минимума (теорема 2.1).

Для доказательства того, что $\mathcal{E} \subset \mathcal{E}'$, нужно установить, что $x+y$ принадлежит $\mathcal{E}'$ и что класс $\mathcal{E}'$ замкнут относительно операций ограниченного суммирования и ограниченного умножения. Заметим, что, согласно следствию 1.2 и теореме 2.2b, класс $\mathcal{E}'$ замкнут относительно операций исчисления высказываний и операций навешивания ограниченных кванторов.

Далее, отношения «меньше чем», «больше чем» и «равно» также являются отношениями класса $\mathcal{E}'$:

$$x \leq y. \equiv .x \div y = 0; \quad x = y. \equiv .x \leq y. \quad x \geq y.$$

¹⁾ Другие интересные определения класса элементарных функций даны Д. Рёдингом в работах [7–9]. — Прим. ред.

Теперь легко видеть, что класс $\mathcal{E}'$ замкнут относительно двух операций максимума, поскольку они определяются посредством операций ограниченного минимума и навешивания ограниченных кванторов. Поэтому следующие функции и отношения принадлежат классу $\mathcal{E}'$:

$$\begin{aligned}x \cdot y &= \mu z \leq (x+1)^{y+1} [(2^x)^y = 2^z], \\x + y &= \mu z \leq (x+1) \cdot (y+1) [2^x \cdot 2^y = 2^z].\end{aligned}$$

Далее, x^+ , x^{++} , $x|y$, $\text{prime}(x)$, $\exp(x, y)$, xNy , p_x суть функции и отношения класса $\mathcal{E}'$, поскольку они определены посредством функций и операций, о которых доказано, что они принадлежат $\mathcal{E}'$; это легко усмотреть из структуры их определений. Отсюда следует, что класс $\mathcal{E}'$ замкнут относительно операции ограниченной рекурсии, так как в доказательстве теоремы 2.3 схема рекурсии была сведена к операциям ограниченного минимума, навешивания ограниченных кванторов и к функциям, описанным выше.

Теперь можно легко доказать с помощью схемы рекурсии, что операции ограниченного суммирования и ограниченного умножения не выводят из класса $\mathcal{E}'$. Именно, предположим, что функции f и g определены через ограниченное суммирование и ограниченное умножение функции F из класса $\mathcal{E}'$ соответственно:

$$f(u, x) = \sum_{i \leq x} F(u, i); \quad g(u, x) = \prod_{i \leq x} F(u, i).$$

Легко доказать, что функции f и g удовлетворяют условиям (a), (b) и (c) операций ограниченной рекурсии:

$$\begin{aligned}f(u, 0) &= F(u, 0), \\f(u, x+1) &= f(u, x) + F(u, x+1), \\f(u, x) &\leq (x+1) \cdot (\text{Max } F(u, z) \text{ при } z \leq x), \\g(u, 0) &= F(u, 0), \\g(u, x+1) &= g(u, x) \cdot F(u, x+1), \\g(u, x) &\leq (\text{Max } F(u, z) \text{ при } z \leq x)^{x+1}.\end{aligned}$$

Итак, функции f и g также принадлежат классу $\mathcal{E}'$. Поэтому класс $\mathcal{E}'$ замкнут относительно всех операций класса $\mathcal{E}$ и включает в себя исходные функции класса $\mathcal{E}$, откуда следует $\mathcal{E} \subset \mathcal{E}'$.

Мы доказали тем самым, что классы $\mathcal{E}$ и $\mathcal{E}'$ совпадают.

Обозначим через $\mathcal{E}''$ наименьший класс функций,

1° включающий $x+1$ и x^y в качестве исходных функций,

2° замкнутый относительно операций подстановки и ограниченной рекурсии.

Теорема 2.5. *Класс $\mathcal{E}''$ тождествен классу $\mathcal{E}$.*

Доказательство. Включение $\mathcal{E}'' \subset \mathcal{E}$ следует из теоремы 2.3. Остается доказать обратное включение $\mathcal{E} \subset \mathcal{E}''$. Мы докажем его косвенным образом, показав, что $\mathcal{E}' \subset \mathcal{E}''$, и используя теорему 2.4.

Легко показать, что оставшаяся исходная функция класса $\mathcal{E}$, а именно $x \dot{-} y$, и следующие функции: $P(x) = x \dot{-} 1$, $x + y$, $x \cdot y$ принадлежат классу $\mathcal{E}''$.

Эти функции удовлетворяют условиям схемы рекурсии:

$$\begin{array}{ll} P(0) = 0, & y \dot{-} 0 = y, \\ P(x+1) = x, & y \dot{-} (x+1) = P(y \dot{-} x), \\ P(x) \leq x, & y \dot{-} x \leq y, \\ y+0 = y, & y \cdot 0 = 0, \\ y+(x+1) = (y+x)+1, & y \cdot (x+1) = y \cdot x + y, \\ y+x \leq (y+1)^{x+1}; & y \cdot x \leq y^x. \end{array}$$

Докажем, что класс $\mathcal{E}''$ замкнут относительно суженной операции ограниченного суммирования: если $F \in \mathcal{E}''$ и $f(u, x) = \sum_{i \leq x} (1 \dot{-} F(u, i))$, то $f \in \mathcal{E}''$, поскольку f можно определить схемой ограниченной рекурсии:

$$\begin{aligned} f(u, 0) &= 1 \dot{-} F(u, 0), \\ f(u, x+1) &= f(u, x) + (1 \dot{-} F(u, x+1)), \\ f(u, x) &\leq x+1. \end{aligned}$$

Таким образом, из следствия 2.1 вытекает, что класс $\mathcal{E}''$ замкнут относительно операции ограниченного минимума. Поэтому $\mathcal{E}' \subset \mathcal{E}''$ и, согласно теореме 2.4, $\mathcal{E} \subset \mathcal{E}''$. Значит, классы $\mathcal{E}$ и $\mathcal{E}''$ тождественны друг другу.

Пусть $\mathcal{E}'''$ обозначает наименьший класс,

1° включающий в качестве исходных функций $x+1$, $x \dot{-} y$, $x \cdot y$, x^{y+1} ,

2° замкнутый относительно операции подстановки и операции ограниченного суммирования.

Теорема 2.6. *Класс $\mathcal{E}'''$ тождествен классу $\mathcal{E}$.*

Доказательство. $\mathcal{E}'''$ содержится в $\mathcal{E}$ по определению. Обратно, $\mathcal{E} \subset \mathcal{E}'''$, так как, согласно следствию 2.1а, $\mathcal{E}'''$ замк-

¹⁾ Функцию $x \cdot y$ из числа исходных можно исключить, ибо $x \cdot y = \left[\sum_{i \leq y} (x \dot{-} (i \dot{-} i)) \right] \dot{-} x$. — Прим. ред.

нут относительно операции ограниченного минимума. Отсюда и по теореме 2.4 $\mathcal{E} = \mathcal{E}'''$.

Доказанные выше теоремы устанавливают, что при базисной операции подстановки и базисе исходных функций $x+1$, $x+y$, $x \dot{-} y$, $x \cdot y$, x^y ограниченные операции суммирования, минимума и рекурсии эквивалентны. Можно также доказать, что каждая из упомянутых операций эквивалентна любой из следующих: операция ограниченного умножения, две операции максимума и операция минимального значения:

$$\text{Min } F(u, x) \text{ (для } x \leq z) = \mu y \leq F(u, 0) \left[\sum_{x \leq z} y = F(u, z) \right].$$

Доказательство этих теорем предоставляется читателю.

Итак, класс элементарных функций может быть определен как содержащий упомянутые выше исходные функции и замкнутый относительно операции подстановки и одной из семи вышеперечисленных операций. Поэтому эти операции будут называться элементарными.

3. КЛАССЫ, ОСНОВАННЫЕ НА ОГРАНИЧЕННОЙ РЕКУРСИИ

Кроме класса $\mathcal{E}$, мы также исследуем другие классы функций, основанные на операции ограниченной рекурсии. С этой целью мы докажем несколько теорем более общего характера.

Будем говорить, что класс функций $\mathcal{X}$ индуктивно определим посредством начальных функций $f_1, \dots, f_n$ и операций $O_1, \dots, O_s$, если $\mathcal{X}$ является наименьшим классом, содержащим функции $f_1, \dots, f_n$ и замкнутым относительно операций $O_1, \dots, O_s$. Будем называть класс $\mathcal{X}$ **индуктивно определимым**, если он индуктивно определим посредством операций, описанных в этой статье.

Теорема 3.1. *Если $\mathcal{X}$ индуктивно определим посредством операций подстановки, а также операций $O_{k_1}, \dots, O_{k_s}$, определенных на функциях одного аргумента и не выводящих за пределы класса функций одного аргумента, и $\mathcal{X}$ включает нумерующие пары функции I, K, L , то класс $\mathcal{X}_1$ ¹⁾ индуктивно определим.*

Для упрощения доказательства предположим, что $\mathcal{X}$ включает исходные функции $f_1(x, y), \dots, f_k(x, y)$ двух аргументов и функции $f_{k+1}(x), \dots, f_{k+t}(x)$ одного аргумента; докажем, что класс $\mathcal{X}_1$ совпадает с классом $\mathcal{X}^*$, являющимся наименьшим классом, включающим исходные функции²⁾ $f_1(Kx, Lx), \dots$

¹⁾ Напомним, что через $\mathcal{X}_1$ обозначается множество функций одного аргумента из класса $\mathcal{X}$. — Прим. перев.

²⁾ Мы предполагаем также, что $I(0, 0) = 0$. С другой стороны, условие $I(Kz, Lz) = z$ в этом доказательстве не используется.

$\dots, f_h(Kx, Lx), f_{h+1}(x), \dots, f_{h+t}(x), Kx, Lx, I(x, x), I(x, 0), I(0, x)$ и замкнутым относительно операций $O_{k_1}, \dots, O_{k_s}$ и O_1, O_2 , где O_1 : если $f(x)$ и $g(x)$ принадлежат $\mathcal{X}^*$, то $f(g(x))$ принадлежит $\mathcal{X}^*$; O_2 : если f принадлежит $\mathcal{X}^*$, то $I(Kx, f(Lx))$ и $I(f(Kx), Lx)$ также принадлежат $\mathcal{X}^{*1}$). Включение $\mathcal{X}^* \subset \mathcal{X}_1$ очевидно. Исходные функции класса $\mathcal{X}^*$ принадлежат классу $\mathcal{X}_1$, а операции, упомянутые в определении класса $\mathcal{X}^*$, не выводят из $\mathcal{X}_1$. Обратное включение будет установлено с помощью некоторых лемм. Если f — функция одного аргумента, то n -я итерация функции f будет обозначаться через f^n . Таким образом, $f^0(x) = x$, $f^{n+1}(x) = f(f^n(x))$. Положим $C_n x = KL^{n-1}x$, откуда $C_1 x = Kx$, $C_2 x = KLx$. Далее,

$$\langle x_1, \dots, x_n, y \rangle = I(x_1, I(x_2, \dots, I(x_n, y) \dots)).$$

Из свойств нумерующих пары функций I, K, L следует, что

$$C_i \langle x_1, \dots, x_n, y \rangle = x_i \text{ для } 0 < i \leq n,$$

$$L^n \langle x_1, \dots, x_n, y \rangle = y, \quad C_n L^m x = C_{n+m} x,$$

$$\langle x_1, \dots, x_n, \langle y_1, \dots, y_m, z \rangle \rangle = \langle x_1, \dots, x_n, y_1, \dots, y_m, z \rangle.$$

Лемма 1. Если $f, g \in \mathcal{X}^*$, то функция $I(f(x), g(x))$ также принадлежит $\mathcal{X}^*$.

Доказательство. Согласно O_2 , функции $I(Kx, g(Lx))$ и $I(f(Kx), Lx)$ принадлежат $\mathcal{X}^*$. Отсюда функции

$$I(f(KI(x, x)), LI(x, x)) = I(f(x), x),$$

$$I(KI(f(x), x), g(LI(f(x), x))) = I(f(x), g(x))$$

также принадлежат $\mathcal{X}^*$ благодаря операции O_1 .

Лемма 2. Если $j_1, \dots, j_k$ принадлежат $\mathcal{X}^*$, то функция $\langle j_k(x), \dots, j_1(x) \rangle$ также принадлежит $\mathcal{X}^*$.

Доказательство. Поскольку $j_2, j_1 \in \mathcal{X}^*$, то, согласно лемме 1, функция

$$\langle j_2(x), j_1(x) \rangle = I(j_2(x), j_1(x))$$

также принадлежит $\mathcal{X}^*$. Предположим, что лемма верна для $k = n$, т. е. что $\langle j_n(x), \dots, j_1(x) \rangle$ принадлежит $\mathcal{X}^*$. По лемме 1 функция

$$\langle j_{n+1}(x), \dots, j_1(x) \rangle = I(j_{n+1}(x), \langle j_n(x), \dots, j_1(x) \rangle)$$

¹⁾ Как сообщил в частном письме автор, операцию O_2 можно заменить более простой O'_2 : если функции $f(x)$ и $g(x)$ принадлежат классу $\mathcal{X}^*$, то $I(f(x), g(x))$ также принадлежит $\mathcal{X}^*$. — Прим. перев.

также принадлежит $\mathcal{X}^*$, так что наша лемма верна для $k = n + 1$. Итак, мы доказали по индукции, что для каждого k функция $\langle j_k(x), \dots, j_1(x) \rangle$ принадлежит $\mathcal{X}^*$.

Лемма 3. Если $j_1, \dots, j_k \in \mathcal{X}^*$, то функции

$$f(x) = \langle j_1(x), \dots, j_k(x), 0 \rangle,$$

$$g(x) = \langle j_1(x), \dots, j_i(x), 0, j_{i+1}(x), \dots, j_k(x), 0 \rangle$$

также принадлежат $\mathcal{X}^*$.

Доказательство. Классу $\mathcal{X}^*$ принадлежит функция $\langle j_k(x), 0 \rangle$ (получается из $I(x, 0)$ подстановкой, т. е. операцией O_1). Отсюда $f(x)$ также принадлежит $\mathcal{X}^*$, и по лемме 2

$$h(x) = \langle j_{i+1}(x), \dots, j_k(x), 0 \rangle = \langle j_{i+1}(x), \dots, j_{k-1}(x), \langle j_k(x), 0 \rangle \rangle \in \mathcal{X}^*.$$

Аналогично получается для $I(0, h(x))$ (из $I(0, x)$) и $g(x) = \langle j_1(x), \dots, j_i(x), I(0, h(x)) \rangle$ также по лемме 2.

Лемма 4. Если $f(x_1, \dots, x_n)$ является функцией класса $\mathcal{X}$, то $f'(x) = f(C_1x, \dots, C_nx)$ является функцией класса $\mathcal{X}^*$.

Доказывается индукцией по порядку функции f в классе $\mathcal{X}^1$.

Если f является исходной функцией двух аргументов класса $\mathcal{X}$, $f(x, y) = f_i(x, y)$, $1 \leq i \leq k$, то функция $f_i(Kx, Lx)$ является исходной функцией класса $\mathcal{X}^*$. Этот класс включает в себя также функции Kx , $I(Kx, KLx)$ (по O_2) и

$$f_i(KI(Kx, KLx), LI(Kx, KLx)) = f_i(Kx, KLx) = f_i(C_1x, C_2x) = f'_i(x).$$

Теперь предположим, что для всех функций g и h порядка ниже, чем n , соответствующие функции g' и h' принадлежат классу $\mathcal{X}^*$ и что f является функцией порядка n . Мы докажем, что f' также принадлежит $\mathcal{X}^*$. Функция f получена из некоторых функций порядка ниже, чем n , посредством операций подстановки или какой-либо из операций $O_{k_1}, \dots, O_{k_s}$. Следовательно, нам нужно рассмотреть следующие случаи:

А. Функция f получена посредством операции подстановки из функций g и h

$$\begin{aligned} f(x_1, \dots, x_k, y_1, \dots, y_m) &= \\ &= g(x_1, \dots, x_n, h(y_1, \dots, y_m), x_{n+1}, \dots, x_k). \end{aligned}$$

¹⁾ По-видимому, здесь автор имеет в виду обычное индуктивное определение порядка функции по глубине построения, при котором исходным функциям приписаны порядки 0. — Прим. ред.

По индуктивному предположению отсюда следует, что функции

$$g'(x) = g(C_1x, \dots, C_{k+1}x),$$

$$h'(x) = h(C_1x, \dots, C_mx)$$

принадлежат $\mathcal{X}^*$. $\mathcal{X}^*$ включает в себя $C_t x$ для любого t . Отсюда, согласно лемме 3, следует, что функция $\langle C_{n+1}x, \dots, C_{n+m}x, 0 \rangle$ также принадлежит $\mathcal{X}^*$. Таким образом, $\mathcal{X}^*$ включает

$$h'(\langle C_{n+1}x, \dots, C_{n+m}x, 0 \rangle) = h(C_{n+1}x, \dots, C_{n+m}x),$$

$$j(x) = \langle C_1x, \dots, C_nx, h(C_{n+1}x, \dots, C_{n+m}x), C_{n+m+1}x, \dots, C_{m+k}x, 0 \rangle$$

(по лемме 3), а также функцию

$$g'(j(x)) = g(C_1x, \dots, C_nx, h(C_{n+1}x, \dots, C_{n+m}x),$$

$$C_{n+m+1}x, \dots, C_{m+k}x) = f'(x).$$

В. f получается из функции g отождествлением k -й и j -й переменных, т. е.

$$f(x_1, \dots, x_n) = g(x_1, \dots, x_j, \dots, x_{k-1}, x_j, x_{k+1}, \dots, x_n).$$

По индуктивному предположению $g' \in \mathcal{X}^*$, тогда $\mathcal{X}^*$ включает также (по лемме 3) функцию

$$j(x) = \langle C_1x, \dots, C_jx, \dots, C_{k-1}x, C_jx, C_{k+1}x, \dots, C_nx, 0 \rangle,$$

а следовательно, и функцию $f'(x) = g'(j(x))$.

С. f получается из g подстановкой константы

$$f(x_1, \dots, x_{k-1}, x_{k+1}, \dots, x_n) =$$

$$= g(x_1, \dots, x_{k-1}, 0, x_{k+1}, \dots, x_n) \text{ и } g' \in \mathcal{X}^*.$$

Положим

$$j(x) = \langle C_1x, \dots, C_{k-1}x, 0, C_{k+1}x, \dots, C_nx, 0 \rangle; \quad j \in \mathcal{X}^*$$

по лемме 3, поэтому функция $f'(x) = g'(j(x))$ также принадлежит $\mathcal{X}^*$.

Д. Если f получается из g и h посредством одной из операций $O_{k_1}, \dots, O_{k_s}$, то f , g и h являются функциями одного аргумента. Имеем

$$g'(x) = g(Kx), \quad h'(x) = h(Kx), \quad g(x) = g'(I(x, x)),$$

$$h(x) = h'(I(x, x)), \quad f'(x) = f(Kx).$$

Поэтому если $g', h' \in \mathcal{X}^*$, то $g, h \in \mathcal{X}^*$ и $f, f' \in \mathcal{X}^*$ в соответствии с определением $\mathcal{X}^*$.

Доказав лемму 4, установим по индукции включение

$$\mathcal{X}_1 \subset \mathcal{X}^*.$$

Если $f(x)$ — функция порядка 0 в классе $\mathcal{X}$, то $f(x) = f_{k+l}(x)$ ($1 \leq l \leq t$) и f принадлежит классу $\mathcal{X}^*$ как исходная функция.

Предположим теперь, что все функции класса $\mathcal{X}_1$, порядок которых в классе $\mathcal{X}$ ниже, чем n , принадлежат $\mathcal{X}^*$. Докажем, что функция $f(x)$ порядка n в классе $\mathcal{X}$ также принадлежит классу $\mathcal{X}^*$.

Будем различать два случая.

А. f получена из функций одного аргумента посредством операций $O_1, O_{k_1}, \dots, O_{k_s}$, откуда $f \in \mathcal{X}^*$, потому что операции $O_1, O_{k_1}, \dots, O_{k_s}$ не выводят из класса $\mathcal{X}^*$.

В. Функция f получена из некоторой функции двух аргументов $F(x, y)$ посредством одной из операций подстановки, т. е.

$$f(x) = F(x, x), \text{ или } f(x) = F(x, 0), \text{ или } f(x) = F(0, x).$$

По лемме 4 функция $F'(x) = F(C_1x, C_2x)$ принадлежит $\mathcal{X}^*$. $\mathcal{X}^*$ включает в себя функции

$$I(Kx, I(Lx, 0)),$$

$$I(KI(x, x), I(LI(x, x), 0)) = \langle x, x, 0 \rangle,$$

а также и функции $\langle x, 0, 0 \rangle$ и $\langle 0, x, 0 \rangle$. Легко показать, что

$$F(x, x) = F'(\langle x, x, 0 \rangle),$$

$$F(x, 0) = F'(\langle x, 0, 0 \rangle),$$

$$F(0, x) = F'(\langle 0, x, 0 \rangle).$$

Таким образом, в каждом возможном случае функция f принадлежит классу $\mathcal{X}^*$. Поэтому классы $\mathcal{X}_1$ и $\mathcal{X}^*$ тождественны.

Теорема 3.2. Класс $\mathcal{E}_1$ индуктивно определим.

Доказательство. Операция ограниченного суммирования может быть определена таким образом, что она функцию $F(x)$ преобразует в функцию $f(x)$:

$$f(x) = \sum_{i \leq Kx} F(I(i, Lx))^{1)}.$$

¹⁾ Введенную ранее операцию ограниченного суммирования можно выразить, например, так: пусть $h(i, x_1, \dots, x_n) \in \mathcal{E}$ ($n > 0$) и $g(y, x_1, \dots, x_n) = \sum_{i \leq y} h(i, x_1, \dots, x_n)$. Положим

$$F(x) = h(c_1x, \dots, c_{n+1}x),$$

$$f(x) = \sum_{i \leq Kx} F(I(i, Lx)).$$

Тогда

$$g(y, x_1, \dots, x_n) = f(\langle y, x_1, \dots, x_n \rangle).$$

— Прим. ред.

Таким образом, из теоремы 3.1 вытекает, что класс $\mathcal{E}_1$ может быть основан на этой операции и на операциях O_1 и O_2 . Класс $\mathcal{E}_k$ также является индуктивно определимым для каждого $k > 0$.

Мы будем говорить, что функция $f(x)$ растет быстрее, чем любая функция класса $\mathcal{E}$, если для всякой функции $g \in \mathcal{E}$ существует такое число n_0 , что для каждого $x \geq n_0$ имеет место неравенство $f(x) > g(x)$.

Функция $f(x_1, \dots, x_n, \dots, x_k)$ называется **неубывающей относительно n -го аргумента**, если для всех $x_1, \dots, x_k$ и y неравенство $x_n \leq y$ влечет

$$f(x_1, \dots, x_n, \dots, x_k) \leq f(x_1, \dots, y, \dots, x_k).$$

Функция, неубывающая относительно каждого своего аргумента, называется просто **неубывающей**. Функция называется **возрастающей**, если она удовлетворяет сформулированному выше условию с заменой неравенства $\leq$ на строгое неравенство $<$.

Далее, будем говорить, что функция g **мажорирует** функцию f , если $f(u) \leq g(u)$ для всех u .

Теорема 3.3. Если классы $\mathcal{E}$ и $\mathcal{U}$ индуктивно определимы посредством операции подстановки и, быть может, посредством операции ограниченной рекурсии и если класс $\mathcal{E}$ включает в себя неубывающие функции, которые мажорируют (все) исходные функции класса $\mathcal{U}$, то для всякой функции f класса $\mathcal{U}$ можно указать некоторую неубывающую функцию f' из класса $\mathcal{E}$, которая мажорирует f и имеет столько же аргументов, сколько их имеет функция f .

Доказательство. Доказательство проводится по индукции относительно порядка функции f в классе $\mathcal{U}$.

Для исходных функций это свойство предполагается (в условии теоремы) выполненным.

Если для любых функций g, h, j порядка ниже, чем n , существуют такие неубывающие функции g', h', j' , что

$$(I) \quad g(x, y) \leq g'(x, y),$$

$$(II) \quad h(y) \leq h'(y),$$

$$(III) \quad j(x, u) \leq j'(x, u),$$

и f — функция порядка n в классе $\mathcal{U}$, то, коль скоро f получена подстановкой функций $f(x, y) = g(x, h(y))$, f мажорируется неубывающей функцией $g'(x, h'(y))$, поскольку из (I), (II) и из факта неубывания g' следует, что

$$f(x, y) = g(x, h(y)) \leq g'(x, h(y)) \leq g'(x, h'(y)).$$

Если, однако, f получена из функции g отождествлением переменных, например x и y , или подстановкой константы, например вместо y , то неубывающая функция f' , полученная из g' применением той же операции, мажорирует f .

Из неравенства (I) следует, что

$$g(y, y) \leq g'(y, y), \quad g(x, 0) \leq g'(x, 0).$$

Наконец, если f получена из функций g, h, j ограниченной рекурсией и условие (с) операции ограниченной рекурсии имеет вид

$$f(x, n) \leq j(x, n),$$

то из неравенства (III) следует, что j' мажорирует f .

Теорема 3.4. Если $\mathcal{X}$ и $\mathcal{Y}$ индуктивно определимы посредством операции подстановки и, кроме того, быть может, посредством операции ограниченной рекурсии, если класс $\mathcal{X}$ включает неубывающие функции, которые мажорируют исходные функции класса $\mathcal{Y}$, и если f растет быстрее любой функции из класса $\mathcal{X}_1$, то f растет быстрее любой функции из класса $\mathcal{Y}_1$.

Доказательство. По теореме 3.3 каждая функция $g \in \mathcal{Y}_1$ мажорируется определенной функцией $g' \in \mathcal{X}_1$, что означает

$$g(x) \leq g'(x)$$

для любого x . Если f растет быстрее, чем g' , то существует такое n , что для $x \geq n$ имеем $g'(x) < f(x)$. Отсюда для $x \geq n$ имеем также неравенство $g(x) < f(x)$, которое означает, что f растет быстрее g .

Теорема 3.5. Если $\mathcal{X}$ и $\mathcal{Y}$ включают функцию следования и индуктивно определимы операциями подстановки и, быть может, другими операциями и если для каждой функции класса $\mathcal{X}_1$ существует функция класса $\mathcal{Y}_1$, которая мажорирует ее, то класс $\mathcal{X}$ не содержит универсальной функции для класса $\mathcal{Y}_1$.

Доказательство. Если $F(n, x)$ — универсальная функция для класса $\mathcal{Y}_1$, тогда если бы класс $\mathcal{X}$ включал в себя $F(n, x)$, то $\mathcal{X}_1$ должен был бы содержать функцию $F(x, x) + 1$. Последняя мажорируется определенной функцией $g \in \mathcal{Y}_1$. Поскольку F — универсальная функция для $\mathcal{Y}_1$, то существует такое n , что $g(x) = F(n, x)$ для каждого x . Подставляя, однако, сюда $x = n$, получаем противоречивое неравенство

$$F(n, n) = g(n) \geq F(n, n) + 1.$$

Аналогично, универсальная функция для класса $\mathcal{X}_n$ не входит в класс $\mathcal{X}$. И все же универсальная функция для класса, который замкнут только относительно операций подстановки, может быть всегда определена с помощью ограниченной рекурсии¹⁾.

Теорема 3.6. Если класс $\mathcal{X}$ содержит нумерующие пары функции I, K, L и индуктивно определим операциями подстановки, а класс $\mathcal{Y}$ индуктивно определим операциями подстановки и ограниченной рекурсии и $\mathcal{Y}$ содержит такую функцию $h(n, x)$, что $f(x) < h(n, x)$, если f имеет порядок n в классе $\mathcal{X}_1$, тогда класс $\mathcal{Y}$ содержит универсальную функцию для класса $\mathcal{X}_1$ ²⁾.

Доказательство. По теореме 3.1 класс $\mathcal{X}_1$ индуктивно определим посредством операций O_1, O_2 .

Функция $F(n, x)$, универсальная для класса $\mathcal{X}_1$, может быть определена следующим образом:

$$(a) \left. \begin{array}{l} F(0, x) = \\ \vdots \\ F(k, x) = \end{array} \right\} \text{исходные функции для класса } \mathcal{X}_1;$$

(b) для $n \geq k$

$$F(n+1, x) = \begin{cases} F(T_1(n), F(T_2(n), x)), & \text{когда } T_3(n) = 0, \\ I(Kx, F(T_1(n), Lx)), & \text{когда } T_3(n) = 1, \\ I(F(T_1(n), Kx), Lx), & \text{когда } T_3(n) > 1; \end{cases}$$

(c) $F(n, x) \leq h(n, x)$.

Случай $T_3(n) = 0$ соответствует операции O_1 .

Случай $T_3(n) = 1$ и $T_3(n) > 1$ — операции O_2 .

Легко заметить, что, обратно, если $\mathcal{Y}$ включает функцию $F(n, x)$, универсальную для $\mathcal{X}_1$, то класс $\mathcal{Y}$ включает также функцию $h(n, x)$, которая обладает свойством, указанным в теореме, а именно

$$h(n, x) = \sum_{i \leq x} F(n, i) + 1.$$

¹⁾ Точный смысл этого высказывания, по-видимому, уточняется в теореме 3.6. — Прим. ред.

²⁾ Теорема и последующее замечание верны, когда $h(n, x)$ монотонна по n , а класс $\mathcal{Y}$ содержит функции, необходимые для сведения построений, приводимых автором ниже, к обыкновенной ограниченной рекурсии. — Прим. ред.

4. КЛАССЫ $\mathcal{E}^n$

Рассмотрим следующую последовательность вычислимых функций:

$$\begin{aligned}f_0(x, y) &= y + 1, \\f_1(x, y) &= x + y, \\f_2(x, y) &= (x + 1) \cdot (y + 1); \\ \text{для } n \geq 2\end{aligned}$$

$$\begin{aligned}f_{n+1}(0, y) &= f_n(y + 1, y + 1), \\f_{n+1}(x + 1, y) &= f_{n+1}(x, f_{n+1}(x, y)).\end{aligned}$$

Докажем некоторые свойства этой последовательности, которые будут нужны в наших дальнейших рассмотрениях.

Теорема 4.1. $f_n(x, y) > y$ для $n > 1$.

Доказательство по индукции.

Если $n=2$, то $f_2(x, y) = (x+1) \cdot (y+1) > y$.

Предположим теперь, что эта теорема справедлива для данного $n > 1$ при любых x и y . Тогда она справедлива также для $n+1$. Действительно, из индуктивного предположения следует, что

$$f_{n+1}(0, y) = f_n(y + 1, y + 1) > y + 1 > y.$$

Предположим, что неравенство $f_{n+1}(x, y) > y$ имеет место для данного x и любого y . Тогда оно выполняется также для $x+1$:

$$f_{n+1}(x + 1, y) = f_{n+1}(x, f_{n+1}(x, y)) > f_{n+1}(x, y) > y.$$

Таким образом, теорема верна для всех x и n .

Теорема 4.2. $f_{n+1}(x + 1, y) > f_{n+1}(x, y)$ для $n \geq 0$.

Доказательство. Для $n \geq 2$ теорема справедлива в силу теоремы 4.1:

$$f_{n+1}(x + 1, y) = f_{n+1}(x, f_{n+1}(x, y)) > f_{n+1}(x, y).$$

Для $n=0, 1$ непосредственно проверяем:

$$x + 1 + y > x + y; \quad (x + 2) \cdot (y + 1) > (x + 1) \cdot (y + 1).$$

Теорема 4.3. $f_n(x, y + 1) > f_n(x, y)$ для $n > 0$.

Доказательство. Для $n = 0, 1$ мы проверяем теорему непосредственно. Для $n \geq 2$ докажем ее по индукции. Предположим, что теорема справедлива для данного n и любых x и y . Отсюда и из теоремы 4.2 получаем

$$\begin{aligned}f_{n+1}(0, y + 1) &= f_n(y + 2, y + 2) > f_n(y + 1, y + 2) > \\ &> f_n(y + 1, y + 1) = f_{n+1}(0, y).\end{aligned}$$

Предположим, что теорема 4.3 верна для данных x и $n+1$ и для любого y . Установим, что она верна также для $x+1$:

$$f_{n+1}(x+1, y+1) = f_{n+1}(x, f_{n+1}(x, y+1)) > f_{n+1}(x, f_{n+1}(x, y)) = f_{n+1}(x+1, y).$$

Теоремы 4.2 и 4.3 означают, что каждая из функций $f_n(x, y)$ для $n > 0$ является строго возрастающей относительно обоих аргументов. Как мы увидим, функции f_n для $n > 3$ не являются элементарными, но они вычислимы. Посредством последовательности f_n можно определить следующую последовательность классов вычислимых функций. Пусть $\mathcal{E}^n$ — наименьший класс,

1) включающий $x+1$, $U_1(x, y) = x$, $U_2(x, y) = y$, $f_n(x, y)$ как исходные функции и

2) замкнутый относительно операций подстановки и ограниченной рекурсии.

Последовательность классов $\mathcal{E}^n$ обладает следующими свойствами.

Теорема 4.4. $\mathcal{E}^3$ является классом элементарных функций.

Доказательство. Функция f_3 является элементарной:

$$f_3(0, y) = (y+2)^2, f_3(1, y) = ((y+2)^2 + 2)^2 \text{ и т. д.}$$

Вообще f_3 легко можно определить с помощью ограниченной рекурсии:

$$g(0, y) = y; g(x+1, y) = (g(x, y) + 2)^2;$$

$$g(x, y) < (y+2)^{2^{2^x}}; f_3(x, y) = g(2^x, y).$$

В классе $\mathcal{E}^3$ также можно определить функцию x^y :

$$x+0 = x; x \cdot 0 = 0 = U_2(x, 0); x^0 = 1;$$

$$x + (y+1) = (x+y) + 1; x(y+1) = xy + x; x^{y+1} = x^y \cdot x;$$

$$x+y \leq f_3(x, y); x \cdot y \leq f_3(x, y); x^y \leq f_3(x, y).$$

Таким образом, исходные функции класса $\mathcal{E}^3$ принадлежат $\mathcal{E}$, и обратно. По теореме 2.5 классы $\mathcal{E}^3$ и $\mathcal{E}$ замкнуты относительно одних и тех же операций. Следовательно, они совпадают.

Теорема 4.5. Класс $\mathcal{E}^0$ включает нумерующие пары функции $Qx = Ex = x \div [\sqrt{x}]^2$, Rx и $Q^n x$ и замкнут относительно операции ограниченного минимума.

Доказательство. Посредством операций класса $\mathcal{E}^0$ можно определить следующие функции:

$$1) O(x) = U_2(x, 0);$$

$$2) U_1(x, y, z) = U_1(x, U_1(y, z));$$

- 3) $U_2(x, y, z) = U_1(U_2(x, y), z);$
 4) $U_3(x, y, z) = U_2(x, U_2(y, z));$
 5) $P(x) = x \div 1: P(0) = O(0),$
 $P(x+1) = U_1(x, P(x)),$
 $P(x) \leq U_1(x, x);$
 6) $x \div y:$ $x \div 0 = U_1(x, x),$
 $x \div (y+1) = U_3(x, y, P(x \div y)),$
 $x \div y \leq U_1(x, y);$
 7) $\sigma(x, y) = x \cdot 0^y = x \cdot (1 \div y): \sigma(x, 0) = U_1(x, x),$
 $\sigma(x, y+1) = O(U_3(x, y, \sigma(x, y))),$
 $\sigma(x, y) \leq U_1(x, y);$
 8) $\tau(x, y) = x + 0^y:$ $\tau(x, 0) = x + 1,$
 $\tau(x, y+1) = U_1(x, y, \tau(x, y)),$
 $\tau(x, y) \leq U_1(x, y) + 1;$
 9) $r(x, y) = \text{остатку от деления } x \text{ на } y:$
 $r(0, y) = O(y),$
 $r(x+1, y) = U_2(x, \sigma(r(x, y) + 1, 1 \div (y \div (r(x, y) + 1))))),$
 $r(x, y) \leq U_2(x, y);$
 10) функция $\left[\frac{x}{y}\right]:$
 $\left[\frac{0}{y}\right] = O(y);$
 $\left[\frac{x+1}{y}\right] = \tau\left(\left[\frac{x}{y}\right], r(x+1, y)\right),$
 $\left[\frac{x}{y}\right] \leq U_1(x, y);$
 11) $[V\bar{x}]:$
 $[V\bar{0}] = 0,$
 $[V\bar{x+1}] = \tau\left([V\bar{x}], ([V\bar{x}] + 1) \div \left[\frac{x+1}{[V\bar{x}] + 1}\right]\right),$
 $[V\bar{x}] \leq U_1(x, x);$
 12) $E(0) = 0,$
 $E(x+1) = \sigma\left(E(x) + 1, \left[\frac{x+1}{[V\bar{x}] + 1}\right] \div [V\bar{x}]\right),$
 $E(x) \leq U_1(x, x);$
 13) $Qx = Ex,$
 14) $Q^0(x) = x,$
 $Q^{n+1}(x) = Q(Q^n(x)),$
 $Q^n(x) \leq U_1(x, n);$
 15) $W(0, y) = O(y),$
 $W(x+1, y) = \tau(W(x, y), 2 \div \tau(1 \div (y \div Q(x)), Q(x) \div y)),$
 $W(x, y) \leq U_1(x, y).$

Функция W удовлетворяет уравнению

$$W(x+1, y) = W(x, y) + 0^{Qx \div y}.$$

Значение функции $W(x, y)$ равно числу таких чисел $s < x$, для которых $Qs = y$ (т. е. избыток s над ближайшим квадратом равен y):

$$Rx = W(x, Qx).$$

Функции Qx , Rx — нумерующие пары функции (отличные от тех, что были определены в § 1). Функция Qx каждое свое значение принимает бесконечно много раз. Функция Rx показывает, сколько существует чисел $s < x$, удовлетворяющих равенству $Qs = Qx$.

Функция $P(x, y)$, которая им соответствует, не принадлежит, конечно, классу $\mathcal{E}^0$.

Предположим, что F принадлежит классу $\mathcal{E}^0$. Следовательно, $\mathcal{E}^0$ включает также функцию, которая определяется следующим образом:

$$\begin{aligned} f(u, 0) &= 1 \div F(u, 0), \\ f(u, x+1) &= \tau(f(u, x), F(u, x+1)), \\ f(u, x) &\leq U_2(u, x+1). \end{aligned}$$

Легко видеть, что

$$f(u, x) = \sum_{i \leq x} 1 \div F(u, i).$$

Отсюда следует, что суженная операция суммирования не выводит из класса $\mathcal{E}^0$. Из следствия 2.1а вытекает, что класс $\mathcal{E}^0$ замкнут относительно операции ограниченного минимума.

Теорема 4.6. *Для каждого n класс $\mathcal{E}^n$ замкнут относительно операции ограниченного минимума. Отношения класса $\mathcal{E}^n$ замкнуты относительно операций навешивания ограниченных кванторов и относительно операций исчисления высказываний.*

Доказательство. Данные выше определения можно повторить для каждого класса $\mathcal{E}^n$, поскольку $\mathcal{E}^0 \subset \mathcal{E}^n$. Отсюда по теореме 2.2b мы заключаем, что каждый класс $\mathcal{E}^n$ замкнут относительно операции навешивания ограниченных кванторов¹⁾. Далее, согласно эквивалентностям

$$\begin{aligned} 1 \div x = 0. &\equiv . \sim (x = 0), \\ \sigma(x, 1 \div y) = 0. &\equiv . x = 0. \vee . y = 0, \end{aligned}$$

¹⁾ Из теоремы 2.2b следует лишь замкнутость относительно операции навешивания ограниченного квантора существования. Замкнутость $\mathcal{E}^n$ относительно операции навешивания ограниченного квантора общности следует из дальнейшего. — *Прим. ред.*

отношения класса $\mathcal{E}^n$ замкнуты относительно операций исчисления высказываний.

Теорема 4.7. $\mathcal{E}^n \subset \mathcal{E}^{n+1}$.

Доказательство¹⁾. Покажем сначала, что для $n > 0$ имеет место следующее неравенство:

$$(\alpha) \quad f_{n+1}(x, y) > f_n(x, y).$$

Для $n=1$ имеем

$$f_2(x, y) = (x+1) \cdot (y+1) > x+y = f_1(x, y).$$

Для $n \geq 2$ мы рассуждаем так: если $x=0$, то выполняется неравенство (α)

$$f_{n+1}(0, y) = f_n(y+1, y+1) > f_n(0, y),$$

поскольку f_n является строго возрастающей функцией (теоремы 4.2 и 4.3).

Предположим, что для каждого y

$$f_{n+1}(k, y) > f_n(k, y).$$

Отсюда ввиду того, что f_{n+1} — строго возрастающая функция, получаем

$$\begin{aligned} f_{n+1}(k+1, y) &= f_{n+1}(k, f_{n+1}(k, y)) > \\ &> f_{n+1}(k, f_n(k, y)) > f_n(k, f_n(k, y)) = f_n(k+1, y), \end{aligned}$$

и по индукции также получаем (α) .

Это неравенство поможет нам доказать, что в классе $\mathcal{E}^{n+1}$ функции f_i (для $i \leq n$) могут быть определены посредством ограниченной рекурсии.

Действительно, из неравенства (α) следует, что для $0 < i \leq n$

$$(\beta) \quad f_i(x, y) < f_{n+1}(x, y).$$

Например, если $n > 1$ (или $n > 2$), то легко можно определить $x+y$ (или $x \cdot y$ и x^y), как мы это сделали в классе $\mathcal{E}^3$. Следовательно, имеют место включения

$$\mathcal{E}^0 \subset \mathcal{E}^1 \subset \mathcal{E}^2 \subset \mathcal{E}^3 \subset \mathcal{E}^{n+1} \quad \text{для } n \geq 3.$$

Предположим теперь, что для $n > 2$ и для $i < n$ функции f_i определены в классе $\mathcal{E}^{n+1}$, и докажем, что в этом случае f_{i+1}

¹⁾ Автор подразумевает нестрогое включение $\mathcal{E}^n \subseteq \mathcal{E}^{n+1}$. Строгое включение будет доказано позже (теорема 4.9). — Прим. ред.

также принадлежит классу $\mathcal{E}^{n+1}$. Функция f_{i+1} удовлетворяет условиям

$$\begin{aligned} f_{i+1}(0, y) &= f_i(y+1, y+1), \\ (\gamma) \quad f_{i+1}(x+1, y) &= f_{i+1}(x, f_{i+1}(x, y)), \\ f_{i+1}(x, y) &\leq f_{n+1}(x, y). \end{aligned}$$

Эти условия определяют функцию f_{i+1} и позволяют вычислять ее значения. Но эта схема не является простой схемой ограниченной рекурсии¹⁾.

Функция f_{i+1} , однако, может быть определена в классе $\mathcal{E}^{n+1}$ посредством операции минимума способом, аналогичным тому, которым функции, удовлетворяющие обычной схеме ограниченной рекурсии (теорема 2.3), определяются посредством операции минимума, а именно с применением последовательности простых чисел p_n . Различие состоит только в том, что здесь мы употребляем двойную последовательность простых чисел, определенных как $p(x, y) = p_{P(x, y)}$, где $P(x, y)$ — функция, нумерующая пары²⁾.

$$\begin{aligned} \text{Полагая } ^3) \quad F(x, y) &= p(x, f_{n+1}(x, y))^{(x+1)} [f_{n+1}(x+1, y+1)]^2, \text{ имеем} \\ f_{i+1}(x, y) &= \mu z \leq f_{n+1}(x, y) \left[\sum_{m \leq F(x, y)} \{z+1 = \right. \\ &= \exp(m, p(x, y)) \cdot \prod_{v \leq m} \{\exp(m, p(0, v)) \neq 0. \rightarrow \exp(m, p(0, v)) = \\ &= f_i(v+1, v+1) + 1\} \cdot \prod_{t, v, w \leq m} \{t \neq 0. t = \\ &= \exp(m, p(w+1, v)) \rightarrow \exp(m, p(w, v)) \neq 0\} \}. \end{aligned}$$

Число m , употребляемое в этом определении, обладает следующим свойством: простое число $p(w, v)$ присутствует в разложении m с положительным показателем, когда $f_{i+1}(w, v)$ необходимо для вычисления значения функции $f_{i+1}(x, y)$ в соответствии с определением (γ) .

Легко также доказать, что если $\exp(m, p(w, u)) \neq 0$, то

$$\exp(m, p(w, u)) = f_{i+1}(w, u) + 1.$$

Отсюда следует, что если $f_i \in \mathcal{E}^{n+1}$, то $f_{i+1} \in \mathcal{E}^{n+1}$ для $i \leq n$. Все функции f_i для $i \leq n$ могут быть последовательно определены в классе $\mathcal{E}^{n+1}$. Таким образом, класс $\mathcal{E}^{n+1}$ включает все

¹⁾ Функция f_{n+1} определима посредством операции ограниченной рекурсии со вставками (entanglement). См. Петер [3, стр. 622].

²⁾ Принадлежащая $\mathcal{E}^{n+1}$. — Прим. ред.

³⁾ Автор сообщил в письме о необходимом исправлении для $F(x, y)$ и для f_{i+1} ; мы приводим исправленный вариант. — Прим. перев.

исходные функции предыдущих классов и, следовательно, включает также все эти классы.

Можно легко показать, что для $n \geq 2$ классы $\mathcal{E}^n$ замкнуты относительно операции ограниченного суммирования. Подобно этому для $n \geq 3$ классы $\mathcal{E}^n$ замкнуты относительно операции умножения и т. д. Каждый последующий класс замкнут относительно большего числа ограниченных операций. Класс $\mathcal{E}^4$ замкнут относительно операции

$$\bigcup_{i \leq x} F(i, u) = F(x, u)^{F(x-1, u) \cdots F(0, u)}.$$

Доказательство этой теоремы предоставляется читателю.

Пусть $\mathcal{W}^n$ является наименьшим классом, который включает функции $I, K, L, x+y, x^2, f_n$ и замкнут относительно операции подстановки. По теореме 3.1 класс $\mathcal{W}_1^n$ индуктивно определим, а именно это будет наименьший класс, включающий функции $x^2, Kx, Lx, f_n(Kx, Lx), I(0, x)^1$ и замкнутый относительно операций: 1) O_1 (суперпозиции) и 2) суммирования функций (т. е. если $f, g \in \mathcal{W}_1^n$, то $f(x) + g(x)$ также принадлежит $\mathcal{W}_1^n$)²⁾.

Теорема 4.8. Если f — функция порядка k в классе $\mathcal{W}_1^n$ (где $n \geq 2$), то

$$(\delta) \quad f(x) < f_{n+1}(k, x).$$

Доказательство основано на теоремах 4.1—4.3. Исходные функции удовлетворяют неравенству (δ) , поскольку

$$x^2 < (x+2)^2 = f_3(0, x) \leq f_{n+1}(0, x); \quad Kx, Lx \leq x < f_n(0, x);$$

$$f_n(Kx, Lx) \leq f_n(x, x) < f_n(x+1, x+1) = f_{n+1}(0, x);$$

$$I(0, x) = \frac{x(x+1)}{2} < (x+2)^2 = f_3(0, x) \leq f_{n+1}(0, x).$$

¹⁾ В оригинале среди исходных функций класса $\mathcal{W}_1^n$ функция $I(0, x)$ не содержится. Необходимость ее введения обусловлена тем, что приводимые автором функции $I(x, y) = (x+y)^2 + x$, $Kx = x \div [\sqrt{x}]^2$, $Lx = [\sqrt{x}] - Kx$ не являются функциями, нумерующими пары (см. сноску на стр. 10). — Прим. ред.

²⁾ Обозначим через $\mathcal{W}_*^n$ наименьший класс функций, содержащий функции $x^2, Kx, Lx, f_n(Kx, Lx), I(0, x)$ и замкнутый относительно суперпозиции (O_1) и суммирования. По определению $\mathcal{W}_*^n \subseteq \mathcal{W}_1^n$. С другой стороны, $I(x, x) = 2x(x+1) = I(0, x) + I(0, x) + I(0, x) + I(0, x)$, $KI(x, x) = x$, $I(x, 0) = \frac{x(x+1)}{2} + x = I(0, x) + x$. Кроме того, если $f \in \mathcal{W}_*^n$ и $g(x) = I(Kx, f(Lx))$ (или $g(x) = I(f(Kx), Lx)$), т. е. g получена из f применением операции O_2 , то $g(x) = I(0, Kx + f(Lx)) + Kx$ (соответственно $g(x) = I(0, f(Kx) + Lx) + f(Kx)$). Таким образом, по теореме 3.1 $\mathcal{W}_1^n \subseteq \mathcal{W}_*^n$, т. е. $\mathcal{W}_1^n = \mathcal{W}_*^n$. — Прим. ред.

Предположим, что g и h — функции порядков l и k соответственно в классе $\mathcal{W}_1^n$ и что они удовлетворяют неравенству (δ) . Пусть $k \geq l$, тогда

$$g(x) < f_{n+1}(k, x), \quad h(x) < f_{n+1}(k, x).$$

Отсюда, поскольку f_{n+1} — строго возрастающая функция, получаем

$$g(h(x)) < f_{n+1}(k, h(x)) < f_{n+1}(k, f_{n+1}(k, x)) = f_{n+1}(k+1, x).$$

Далее, применяя неравенство (α) теоремы 4.7, получаем

$$g(x) + h(x) < 2f_{n+1}(k, x) < (f_{n+1}(k, x) + 2)^2 =$$

$$= f_3(0, f_{n+1}(k, x)) \leq f_{n+1}(k+1, x).$$

Поэтому если f — функция порядка $k+1$, полученная из функций g и h посредством одной из операций 1) или 2), то

$$f(x) < f_{n+1}(k+1, x).$$

Таким образом, мы доказали теорему по индукции.

Теорема 4.9. *Функция $f_{n+1}(x, x)$ растет быстрее, чем любая функция класса $\mathcal{E}_1^n$.*

Доказательство. Для $n \geq 2$ из теоремы 4.8 следует что если f — функция порядка k в классе $\mathcal{W}_1^n$ и $x \geq k$, то

$$f(x) < f_{n+1}(x, x).$$

Отсюда $f_{n+1}(x, x)$ растет быстрее любой функции класса $\mathcal{W}_1^n$ и в соответствии с теоремой 3.4 растет быстрее, чем любая функция класса $\mathcal{E}_1^n$, так как исходные функции класса $\mathcal{E}^n$ мажорируются возрастающей функцией $f_n(x+1, y+1)$ класса $\mathcal{W}^n$. Для $n=0, 1$ можно непосредственно проверить справедливость нашей теоремы. Легко можно доказать, что если f имеет порядок k в классе $\mathcal{E}^0$, то $f(x) < x + 2^k + 1$. Аналогично, если f порядка k в классе $\mathcal{E}^1$, то $f(x) < (x+1) \cdot 2^k$.

Отсюда функция $2x$ растет быстрее любой функции $f \in \mathcal{E}^0$ и $(x+1)^2$ растет быстрее, чем всякая функция f класса $\mathcal{E}^1$. Таким образом, мы получили, что $\mathcal{E}^n \subset \mathcal{E}^{n+1}$, $\mathcal{E}^n \neq \mathcal{E}^{n+1}$.

Теорема 4.10. *Класс отношений $\mathcal{E}^n$ для $n > 2$ является наименьшим классом, включающим исходные отношения*

$$R_1(x) \equiv x = 0, \quad R_2(x, y) \equiv x = y + 1, \quad R_3(x, y, z) \equiv x = y^z,$$

$$R_4(x, y, z) \equiv x = y \div z, \quad R_5(x, y, z) \equiv x = f_n(y, z),$$

$$R_6(x, y, z) \equiv x \leq f_n(y+1, z+1), \quad R_6'(x, y) \equiv R_6(x, y, y)$$

и замкнутым относительно операций исчисления высказываний и навешивания ограниченных кванторов.

Доказательство. Под операцией навешивания ограниченного квантора существования мы подразумеваем операцию, которая преобразует отношение U в отношение V , определенное следующим образом:

$$V(x, y, u) \equiv \sum_z R_6(z, x, y) \cdot U(x, y, z, u).$$

В частности, отношение V может иметь одно переменное.

Пусть $\mathcal{E}_*^n$ обозначает наименьший класс отношений, включающий описанные выше исходные отношения и замкнутый относительно определенной только что операции. Из доказанной перед этим теоремы прямо следует, что $\mathcal{E}_*^n$ включает класс отношений класса $\mathcal{E}^n$.

Для доказательства обратного включения нам понадобятся некоторые утверждения, доказанные ниже.

Будем говорить, что выражение $A(x, y_1, \dots, y_m)$ **оценивает x посредством $y_1, \dots, y_m$** при условии, что выражение A имеет вид

$$\sum_{z_1, z_2, \dots, z_k} R_6(x, z_1, z_2) \cdot R_6(z_1, z_3, z_4) \dots R_6(z_k, y_l, y_j)$$

и каждое из переменных $x, z_1, \dots, z_k$ оказывается первым аргументом в одном из выражений, охватываемых отношением R_6 . Кроме того, переменное x представлено лишь в одном отношении R_6 , и если выражение $R_6(z_i, z_j, z_h)$ представлено в выражении A , то $i < j \leq k$ и $i < h \leq k$.

Очевидно, если выражение $A(x, y_1, \dots, y_m)$ оценивает x посредством $y_1, \dots, y_m$ и $B(x, y_1, \dots, y_m, u)$ является отношением класса $\mathcal{E}_*^n$, то выражение

$$\sum_x A(x, y_1, \dots, y_m) \cdot B(x, y_1, \dots, y_m, u)$$

также является отношением класса $\mathcal{E}_*^n$.

Лемма 4.11. Для каждой функции $f(y_1, \dots, y_m)$ класса $\mathcal{E}^n$ существует оценивающее выражение $A(x, y_1, \dots, y_m)$, такое, что имеет место импликация

$$x \leq f(y_1, \dots, y_m) \rightarrow A(x, y_1, \dots, y_m).$$

Доказательство. $R_6(x, y_1, y_2)$ является оценивающим выражением для исходных функций класса $\mathcal{E}^{n-1}$). Если функ-

¹⁾ Для функции $y_1 + 1$ это означает следующее: если $x \leq y_1 + 1$, то для всех y , $R_6(x, y_1, y)$. — Прим. ред.

ции g и h имеют оценивающие выражения A и B , удовлетворяющие следующим импликациям:

$$\begin{aligned}x &\leq g(y_1, \dots, y_k) \rightarrow A(x, y_1, \dots, y_k), \\x &\leq h(y, u) \rightarrow B(x, y, u),\end{aligned}$$

тогда выражение $\sum_z B(x, z, u) \cdot A(z, y_1, \dots, y_k)$ эквивалентно оценивающему выражению для функции

$$f(y_1, \dots, y_k, u) = h(g(y_1, \dots, y_k), u),$$

поскольку импликация

$$x \leq h(g(y_1, \dots, y_k), u) \rightarrow \sum_z B(x, z, u) \cdot A(z, y_1, \dots, y_k)$$

следует из импликаций, приведенных выше. Оставшиеся операции: отождествление переменных, подстановки констант и ограниченной рекурсии — не представляют никаких трудностей.

Доказав лемму 4.11, мы в свою очередь докажем по индукции, что для каждой функции $f \in \mathcal{E}^n$ при $n > 2$ отношение $x = f(y_1, \dots, y_m)$ принадлежит классу $\mathcal{E}_*^n$. Для исходных функций соответствующие отношения являются исходными в классе $\mathcal{E}_*^n$. Предположим, что для функций g и h соответствующие отношения G и H принадлежат классу $\mathcal{E}_*^n$:

$$\begin{aligned}x = g(y, u) &\equiv G(x, y, u), \\x = h(y_1, \dots, y_m) &\equiv H(x, y_1, \dots, y_m);\end{aligned}$$

тогда получаем эквивалентность

$$\begin{aligned}x = g(h(y_1, \dots, y_m), u) &\equiv \\&\equiv \sum_z \{A(z, y_1, \dots, y_m) \cdot H(z, y_1, \dots, y_m) \cdot G(x, z, u)\},\end{aligned}$$

где A является оценивающей функцией для функции h . Таким образом, это отношение соответствует функции, полученной подстановкой. Далее

$$\begin{aligned}x = g(0, u) &\equiv \sum_z \{R_6(z, x, x) \cdot R_1(z) \cdot G(x, z, u)\}, \\x = g(y, y) &\equiv \sum_z \{R_6(z, y, y) \cdot z = y \cdot G(x, z, y)\}, \\x = \mu v \leq y [g(v, u) = 0] &\equiv \sum_z \{R_6(z, x, x) \cdot R_1(z) \cdot x \leq y \cdot G(x, z, u) \\&\quad \cdot \prod_v \{R_6(v, x, x) \cdot v < x \rightarrow \sim(G(z, v, u))\} \cdot \vee \cdot R_1(x) \\&\quad \cdot \prod_v \{R_6(v, y, y) \cdot v \leq y \rightarrow \sim(G(z, v, u))\}\}.\end{aligned}$$

Отношения $\leq$, $=$, $<$ могут быть определены в $\mathcal{E}_*^n$ посредством отношения R_4 .

Поскольку для $n > 2$ класс $\mathcal{E}^n$ может быть определен как класс, замкнутый относительно операций подстановки и ограниченного минимума, мы доказали, что для каждой функции $f \in \mathcal{E}^n$ отношение $x = f(y_1, \dots, y_m) \equiv F(x, y_1, \dots, y_m)$ принадлежит $\mathcal{E}_*^n$. Отсюда $\mathcal{E}_*^n$ включает также отношение

$$f(y_1, \dots, y_m) = 0. \equiv \sum_z \{R_6(z, y_1, y_1) \cdot R_1(z) \cdot F(z, y_1, \dots, y_m)\}.$$

Таким образом, класс $\mathcal{E}_*^n$ включает все отношения класса $\mathcal{E}^n$.

Теорема 4.12. Для $n > 2$ класс $\mathcal{E}^{n+1}$ включает универсальную функцию для класса $\mathcal{E}_1^n$.

Доказательство. Если $n > 2$, то класс $\mathcal{E}_1^n$ является наименьшим классом, включающим некоторые исходные функции и замкнутым относительно операций суперпозиции, суммирования функций $f(x) + g(x)$ и ограниченного минимума функций одного аргумента

$$h(x) = \mu z \leq Kx [g(I(z, Lx)) = 0].$$

Отсюда функцию $F(n, x)$, универсальную для $\mathcal{E}_1^n$, можно определить следующим образом:

$$\left. \begin{array}{l} F(0, x) = \\ \vdots \vdots \vdots \vdots \vdots \vdots \\ F(k_0, x) = \end{array} \right\} \text{исходные функции для } \mathcal{E}_1^n;$$

а для $n > k_0$

$$F(n+1, x) = \begin{cases} F(T_1 n, F(T_2 n, x)), & \text{когда } T_3 n = 0, \\ F(T_1 n, x) + F(T_2 n, x), & \text{когда } T_3 n = 1, \\ \mu z \leq Kx [F(T_1 n, I(z, Lx)) = 0], & \text{когда } T_3 n > 1. \end{cases}$$

Эти рекурсивные определения могут быть приведены к определениям, сформулированным с помощью ограниченного минимума; это можно сделать посредством двойной последовательности простых чисел $p(x, y)$, как мы это сделали при доказательстве теоремы 4.7. Числа, существование которых предполагается в определениях этого типа, можно легко оценить сверху с помощью функции f_{n+1} . Завершение доказательства предоставляется читателю.

Обозначим через $\mathcal{R}$ класс примитивно рекурсивных функций.

Теорема 4.13. Класс $\mathcal{R}$ является объединением классов $\mathcal{E}^n$.

Доказательство. Заметим, что $f_0, f_1, f_2 \in \mathcal{R}$, и если $f_n \in \mathcal{R}$, то $f_{n+1} \in \mathcal{R}$ (в соответствии с теоремой, доказанной Р. Петер [3]), так как f_{n+1} определяется посредством операции рекурсии со вставками, которая не выводит из класса $\mathcal{R}$.

Отсюда для каждого n $f_n \in \mathcal{R}$. Класс $\mathcal{R}$, будучи замкнут относительно примитивной рекурсии, замкнут и относительно ограниченной рекурсии. Отсюда $\sum_n \mathcal{E}^n \subset \mathcal{R}$.

Лемма 4.14. Если f является функцией порядка не выше n в классе $\mathcal{R}_1$, то $f \in \mathcal{E}^{n+3}$.

Доказательство. Мы исследуем класс $\mathcal{R}_1$, как он определен Р. Робинсоном [5, теорема 3, стр. 940]. Исходные функции класса $\mathcal{R}_1$, а именно $x+1$ и Ex , принадлежат $\mathcal{E}^3$.

Предположим, что функции g и h порядка не выше k удовлетворяют лемме: $g, h \in \mathcal{E}^{k+3}$. Тогда $g(x) + h(x)$ и $g(h(x))$ также принадлежат классу $\mathcal{E}^{k+3}$, поскольку каждый класс $\mathcal{E}^{n+3}$ включает функцию $x+y$ и замкнут относительно операции подстановки. Тем более эти функции принадлежат классу $\mathcal{E}^{k+1+3}$, который включает функцию $h^x(0)$. Именно, если $h \in \mathcal{E}^{k+3}$, то по теореме 3.3 существует некоторая неубывающая функция $h' \in \mathcal{W}^{k+3}$, такая, что h' мажорирует h , потому что класс $\mathcal{W}^{k+3}$ включает возрастающую функцию $f_{k+3}(x+1, y+1)$, которая мажорирует исходные функции класса $\mathcal{E}^{k+3}$. Пусть h' является функцией порядка l в классе $\mathcal{W}^{k+3}$. По теореме 4.8 для каждого x справедливо неравенство

$$(\delta) \quad h(x) \leq h'(x) < f_{k+4}(l, x).$$

Отсюда имеет место неравенство

$$(\epsilon) \quad h^{(y)}(0) < f_{k+4}(l+y, 0).$$

Это неравенство удовлетворяется для $y=0$, поскольку

$$h^{(0)}(0) = 0 < 1 = f_2(0, 0) < f_{k+4}(l, 0).$$

Предположим, что неравенство (ϵ) имеет место для y . Тогда согласно (δ) неравенство (ϵ) выполняется также для $y+1$:

$$\begin{aligned} h^{(y+1)}(0) &= h(h^{(y)}(0)) < f_{k+4}(l, h^{(y)}(0)) < f_{k+4}(l, f_{k+4}(l+y, 0)) \leq \\ &\leq f_{k+4}(l+y, f_{k+4}(l+y, 0)) = f_{k+4}(l+(y+1), 0). \end{aligned}$$

Тем самым мы доказали по индукции неравенство (ϵ) для всех y . Отсюда если $f(x) = h^{(x)}(0)$, то функция f может быть

определена в классе $\mathcal{E}^{k+4}$ посредством ограниченной рекурсии:

$$f(0) = 0, \quad f(x+1) = h(f(x)), \quad f(x) \leq f_{k+4}(l+x, 0).$$

Таким образом, если f имеет в классе $\mathcal{R}_1$ порядок $k+1$, то $f \in \mathcal{E}^{k+1+3}$.

Таким образом, рекурсией по k мы доказали лемму 4.14.

Из леммы 4.14 прямо следует, что каждая функция f класса $\mathcal{R}$ принадлежит некоторому классу последовательности $\mathcal{E}^n$. Пусть f является функцией k аргументов $f(x_1, \dots, x_k)$ и принадлежит классу $\mathcal{R}$, тогда мы определим функцию f' как

$$f'(x) = f(C_1x, C_2x, \dots, C_kx).$$

Функция f' принадлежит $\mathcal{R}_1$. Пусть f' является функцией порядка n в классе $\mathcal{R}_1$. Тогда из леммы 4.14 следует, что $f' \in \mathcal{E}^{n+3}$. Класс $\mathcal{E}^{n+3}$ также включает функцию $f(x_1, \dots, x_k) = f'(\langle x_1, \dots, x_k, 0 \rangle)$.

Будем говорить, что операции (совокупность операций) $O_{k_1}, \dots, O_{k_s}$ являются недостаточными для получения класса $\mathcal{R}$, если для каждого конечного числа функций $g_1, \dots, g_k$, принадлежащих классу $\mathcal{R}$, существует такая функция f класса $\mathcal{R}$, которая не может быть получена посредством операций $O_{k_1}, \dots, O_{k_s}$ при исходных функциях $g_1, \dots, g_k$.

Теорема 4.15. *Операции подстановки и ограниченной рекурсии недостаточны для получения всех примитивно рекурсивных функций.*

Доказательство. Если $g_1, \dots, g_k$ принадлежат $\mathcal{R}$, то из теоремы 4.13 следует, что существует такое n , что все функции $g_1, \dots, g_k$ принадлежат $\mathcal{E}^n$, потому что классы $\mathcal{E}^n$ образуют возрастающую последовательность (теорема 4.7). Поскольку операции ограниченной рекурсии и подстановки не выводят из класса $\mathcal{E}^n$, то эти операции недостаточны для получения функции f_{n+1} , которая принадлежит классу $\mathcal{E}^{n+1}$.

В связи с классами $\mathcal{E}^n$ возникают перечисленные ниже проблемы.

1. Являются ли операции ограниченной рекурсии и ограниченного минимума эквивалентными при наличии операции подстановки и конечного числа исходных функций, принадлежащих классам $\mathcal{E}^0$ (или $\mathcal{E}^1(\mathcal{E}^2)$)?

2. Аналогичная проблема, относящаяся к операциям ограниченного суммирования и ограниченной рекурсии.

3. Может ли универсальная функция для класса $\mathcal{E}_1^0$ принадлежать классам $\mathcal{E}^1$ или $\mathcal{E}^2$?

4. Можно ли характеризовать класс отношений $\mathcal{E}^n$ для $n < 3$ подобно тому, как характеризуется класс отношений $\mathcal{E}^n$ для $n \geq 3$ (теорема 4.10)? Можно ли аналогичным образом охарактеризовать отношения одного аргумента класса $\mathcal{E}^n$?

5. Эквивалентна ли операция, которая описана на стр. 39 (перед теоремой 4.8) и относительно которой замкнуты классы $\mathcal{E}^n$ для $n > 3$, операции ограниченной рекурсии?

6. Выводит ли из класса $\mathcal{E}^n$ операция ограниченной двойной рекурсии¹⁾?

7. Определим следующую последовательность функций:

$$f'_0(x, y) = y + 1, \quad f'_1(x, y) = x + y;$$

для $n \geq 1$

$$f'_{n+1}(0, y) = 1, \quad f'_{n+1}(x + 1, y) = f'_n(f'_{n+1}(x, y), y).$$

Пусть $\mathcal{F}^n$ обозначает наименьший класс, включающий исходные функции $x + 1$, $U_1(x, y)$, $U_2(x, y)$, $f'_n(x, y)$ и замкнутый относительно операций подстановки и ограниченной рекурсии. Можно ли для классов $\mathcal{F}^n$ доказать теоремы²⁾, аналогичные теоремам для классов $\mathcal{E}^n$?

8. Являются ли операции³⁾ подстановки достаточными для получения класса $\mathcal{E}^n$?

9. Обозначим через $\mathcal{R}^2$ класс функций, полученных посредством двойной рекурсии. Является ли операция единичной (single) рекурсии достаточной для получения класса $\mathcal{R}^2$? Аналогичный вопрос для других классов кратно рекурсивных функций⁴⁾.

5. ПРИЛОЖЕНИЯ КЛАССА $\mathcal{E}^0$

Функции класса $\mathcal{E}^0$ могут быть использованы в канонической форме вычислимых функций. Пусть $\iota x[R(\dots x\dots)]$ обозначает единственное x , такое, что $R(\dots x\dots)$.

Теорема 5.1. *Каждая вычислимая функция может быть представлена в виде $f(u) = A(\iota x[B(u, x) = 0])$, где A и B являются функциями класса $\mathcal{E}^0$.*

¹⁾ С. С. Марченков установил, что замыкание класса $\mathcal{E}^n$ ($n \geq 2$) относительно ограниченной двойной рекурсии совпадает с классом $\mathcal{E}^{n+1}$ всех функций из $\mathcal{E}^{n+1}$, ограниченных функциями из $\mathcal{E}^n$ (см. стр. 130 настоящего сборника). — *Прим. перев.*

²⁾ Утвердительный ответ дал Ричи [10] для $\mathcal{E}^n$, $n \geq 2$. — *Прим. ред.*

³⁾ Эта проблема решена в положительном смысле для $n \geq 3$ Д. Рёдингом [7] и для $n = 2$ С. С. Марченковым. — *Прим. ред.*

⁴⁾ См. стр. 131, где дан отрицательный ответ на этот вопрос. — *Прим. перев.*

Из равенства (I) следует, что

$$h(v) = \iota z [S(z, v)]$$

есть функция возрастающая. Отсюда следует, что

$$\begin{aligned} f^{-1}(y) &= \mu t [f(t) = y] = \mu t [AR(\iota z [S(z, t)]) = y] = \\ &= Q\mu s [S(Rs, Qs). ARR s = y], \end{aligned}$$

потому что наименьшее $s = P(z, t)$ соответствует наименьшему t , если число $z = \iota z [S(z, t) = 0] = h(t)$ возрастает с ростом t .

Поскольку, согласно теоремам 4.5 и 4.6, отношение S принадлежит классу $\mathcal{E}^0$, то класс $\mathcal{E}^0$ содержит такое отношение T , что

$$f^{-1}(y) = Q\mu s [T(s, y)].$$

Это означает, что

$$f^{-1}(y) = Q\iota s \left[T(s, y). \prod_{u < s} \sim (T(u, y)) \right].$$

Отношение под знаком ι -операции в этом случае также принадлежит классу $\mathcal{E}^0$, поскольку операции навешивания ограниченных кванторов не выводят из класса $\mathcal{E}^0$.

Операции отождествления переменных не вызывают никаких трудностей.

Теорема 5.2. *Каждая вычислимая функция может быть представлена в форме*

$$f(u) = Q(\iota x [A(x, u) = 0]),$$

называемой канонической формой [6, стр. 716; 2, стр. 727], где Q и A принадлежат классу $\mathcal{E}^0$ и Q является фиксированной функцией, определяющей первый элемент пары.

Доказательство. Нам известно из предыдущей теоремы, что f может быть выражена как

$$f(u) = A(\iota x [B(x, u) = 0]),$$

и отсюда

$$f(u) = Q(\iota x [Qx = ARx. B(Rx, u) = 0]).$$

Теорема 5.3. *Каждое рекурсивно перечислимое множество перечислимо некоторой функцией класса $\mathcal{E}^0$.*

Доказательство. Существует элемент a , такой, что $a \in X$. Множество X перечисляется некоторой функцией f , которая может быть задана в следующей канонической форме:

$$f(v) = Q(\mu x [F(x, v) = 0]).$$

Q и F принадлежат $\mathcal{E}^0$ в соответствии с теоремой 5.2.

Имеют место следующие эквивалентности:

$$\begin{aligned} u \in X. &\equiv \sum_v u = f(v). \equiv \sum_{v, x} u = Qx. x = \mu y [F(y, v) = 0]. \equiv \\ &\equiv \sum_{v, x} u = Qx. F(x, v) = 0. \prod_{y < x} F(y, v) \neq 0. \equiv \\ &\equiv \sum_{v, x} H(u, v, x) = 0. \equiv \sum_y H(u, Qy, Ry) = 0, \end{aligned}$$

где H — некоторая функция, принадлежащая $\mathcal{E}^0$ в силу теоремы 4.5. Из приведенных выше эквивалентностей следует, что если множество X бесконечно, то оно перечисляется функцией

$$j(y, z) \equiv \mu u \leq z + a [H(u, Qy, Ry) = 0],$$

а следовательно, и функцией $g(v) = j(Qv, Rv)$. Эти функции принадлежат классу $\mathcal{E}^0$, потому что этот класс замкнут относительно операции ограниченного минимума и $z + a = z + 1^{(1)} + \dots + 1^{(a)}$, где a — константа.

ЛИТЕРАТУРА¹⁾

1. Gödel K., Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I, *Monatshefte für Math. und Phys.*, **38** (1931), 173—198.
2. Kleene S. C., General recursive Functions of Natural Numbers, *Math. Annalen*, **112** (1936), 727—742.
3. Peter R., Über den Zusammenhang der verschiedenen Begriffe der rekursiven Funktionen, *Math. Annalen*, **110** (1934), 612—632.
4. Peter R., *Rekursive Funktionen*, Budapest, 1951. (Русский перевод: Петер Р., *Рекурсивные функции*, ИЛ, М., 1954.)
5. Robinson R. M., Primitive recursive Functions, *Bull. Amer. Math. Soc.*, **53** (1947), 915—942.
6. Robinson J., General recursive Functions, *Proc. Amer. Math. Soc.*, **1** (1950), 703—718.
- 7*. Rödding D., Über die Eliminierbarkeit von Definitionsschemata in der Theorie der rekursiven Funktionen, *Z. math. Logik und Grunlag. der Math.*, **10**, № 4 (1964), 315—330.
- 8*. Rödding D., Darstellung der (in Kalmar-Csillag'schen Sinne) elementaren Funktionen, *Arch. math. Logik und Grundlagenforsch.*, **7**, № 3—4 (1965), 139—158.
- 9*. Rödding D., Über Darstellung der elementaren Funktionen, II, *Arch. math. Logik und Grundlagenforsch.*, **9**, № 1—2 (1967), 36—48.
- 10*. Ritchie R. W., Classes of recursive functions based on Ackermann's functions, *Pacif. J. Math.*, **15**, № 3 (1965), 1027—1044.

¹⁾ Звездочкой отмечена литература, добавленная при переводе. — Прим. ред.

КЛАССЫ ПРЕДСКАЗУЕМО ВЫЧИСЛИМЫХ ФУНКЦИЙ¹⁾

Р. В. Ричи

ВВЕДЕНИЕ

В этой работе мы изучаем последовательность классов вычислимых функций, сложность вычисления которых можно сравнительно просто предсказать. Класс рекурсивных функций состоит из таких функций, для которых существуют механические процедуры для получения значений из аргументов. Однако этот класс, так как он включает все вычислимые функции, должен содержать функции со сколь угодно сложными вычислениями. Существуют различные более ограниченные классы функций, которые легко вычисляются каким-либо способом, однако это в большой степени происходит за счет интуиции. Приятное исключение составляет класс F — функций, вычислимых²⁾ конечными автоматами. Но этот класс слишком ограничен: он не содержит даже умножения. Мы строим иерархию более содержательных классов функций; каждый из них определяется как класс таких функций, сложность вычисления которых «предсказывается» функцией из предыдущего класса.

Известно, что класс F числовых функций, вычислимых конечными автоматами, содержит сложение и замкнут относительно явных преобразований³⁾ и композиции. В частности, он содержит линейные функции⁴⁾. Этот класс F берется в качестве исходного класса, так как эти функции вычисляются наиболее просто: по входу сразу печатается выход. Для получения большего класса рассматриваем класс функций, вычислимых (в двоичной системе счисления) на машинах Тьюринга, использующих ленту, длина которой ограничена более простой функ-

¹⁾ Ritchie R. W., Classes of predictably computable functions, *Trans. Amer. Math. Soc.*, 106 (1963), 139—173.

²⁾ Эти функции закодированы в общепринятой двоичной системе счисления, но ясно, что в равной степени приемлемо кодирование в n -ичной системе счисления для любого n . Определение F смотри в приложении 2.

³⁾ В смысле Смюльяна. Определение смотри ниже в сноске 1 на стр. 53.

⁴⁾ Включение линейных функций подразумевается в смысле [2, стр. 31 и 41], и этого достаточно для дальнейшего. (Фактически, если читатель хочет, он может без ограничения общности считать F на протяжении всей статьи классом линейных функций.) Свойства F , которые мы будем использовать, исследуются в приложении 2.

цией, именно функцией из F . Обозначим этот новый класс через F_1 . Продолжая таким образом, определяем F_i как класс функций, вычислимых на машинах Тьюринга, которые используют ленту длиной, ограниченной функцией из F_{i-1} . Таким образом, мы измеряем сложность вычисления на машине Тьюринга длиной ленты, необходимой для этого вычисления. (Отсюда мы, конечно, можем получить и некоторые границы для числа шагов в вычислении.)

Мы можем интерпретировать счетное объединение $\mathcal{F}$ классов F_i следующим образом.

Каждая функция, вычислимая на машине, у которой объем памяти, необходимой для вычисления, есть функция, допускающая предсказуемое вычисление (на машине, требующей меньше памяти, чем данная) с помощью функции F , или функция, ограничивающая вычисление, в свою очередь имеет предсказуемое вычисление и т. д., должна принадлежать $\mathcal{F}$. Обратно, каждый элемент $\mathcal{F}$ содержится в некотором F_i и, значит, должен вычисляться в этом смысле не более чем с i предсказаниями. Таким образом, мы говорим об $\mathcal{F}$ как о классе предсказуемо вычислимых функций и рассматриваем классы F_i как образующие разбиение по сложности класса $\mathcal{F}$ всех предсказуемо вычислимых функций.

Исследуя $\mathcal{F}$, мы обнаруживаем сначала, что каждый индивидуальный класс F_i содержит тождественную функцию и функцию следования. Затем мы показываем, что он содержит также функции f_i , где $f_1(x)$ есть 2^x и $f_{i+1}(x)$ определяется как $f_1(f_i(x))$. Далее мы показываем, что каждый класс F_i замкнут относительно явных преобразований и что объединение всех F_i замкнуто относительно композиции и ограниченной рекурсии¹⁾. Комбинируя эти факты, получаем, что класс $\mathcal{E}$ элементарных функций (Чиллаг — Кальмар, см. [4, 6]) содержится в $\mathcal{F}$. Затем доказываем обратное включение, показывая тем самым, что элементарные функции в точности совпадают с предсказуемо вычислимыми функциями.

Последние три раздела этой статьи посвящаются более детальному анализу классов F_i , для чего требуется новая арифметизация машин Тьюринга, которая и проводится в приложении 3. В этих заключительных разделах показывается, что каждый из классов F_i включает класс $\mathcal{E}^2$, определенный Гжегорчиком [3], и каждый класс F_i можно определить без ссылки на машины Тьюринга.

В последнем разделе показывается, что последовательность классов отношений, характеристические функции которых лежат

¹⁾ См. лемму 4 и сноску 1 на стр. 58.

в F_i , образует подразбиение класса всех элементарных отношений. Таким образом, идеи, развитые в этой статье, дают классификацию по сложности вычисления не только для функций, но и для отношений и разрешающих процедур.

1. ОПРЕДЕЛЕНИЕ КЛАССОВ F_i

В терминах обычных определений машин Тьюринга [1, 4] трудно доказать теорему 1, сформулированную ниже, почти в точности повторив стандартные рассуждения из [4, стр. 330, 331]. Единственная модификация, необходимая для того, чтобы избежать слишком расточительного расхода ленты, заключается в следующем: все записи, которые уже больше не будут использоваться, стираются, а образовавшиеся пробелы уничтожаются. Поскольку это достаточно очевидно, мы опускаем детали и будем довольствоваться лишь наброском доказательства теоремы.

Определим *размер ленты, использованной при вычислении* $X_1, \dots, X_q$, как максимум длин лент t_i , встречающихся в мгновенных описаниях X_i . Будем обозначать размер ленты, использованной при вычислении $f(x_1, \dots, x_n)$, через ¹⁾ $a_f(x_1, \dots, x_n)$, а длину ленты, на которой записаны числа $x_1, \dots, x_n$, через $l(x_1, \dots, x_n)$. Обозначая сокращенно $x_1, \dots, x_n$ через x , получаем следующие результаты:

Теорема 1.(а) Если h и g вычислимы и f определяется посредством равенства

$$f(x, y) = h(x, g(y)),$$

то

$$a_f(x, y) = l(x, y) + \max[l(x) + a_g(y), a_h(x, g(y))],$$

где $l(x) = \sum_{i=1}^n l(x_i) + (n-1)$, так как, записывая на ленте $x_1, \dots, x_n$, мы разделяем их пробелами.

(б) Если h, g вычислимы и f определяется из них примитивной рекурсией, т. е. $f(x, 0) = g(x)$ и $f(x, s(y)) = h(x, y, f(x, y))$, где $s(y)$ — функция следования ($y + 1$), то

$$a_f(x, y) = 2 \left[l(x, y) + \max_{z < y} [\max(a_h(x, z, f(x, z))), a_g(x)] \right].$$

¹⁾ Это обозначение конечно, некорректно, так как a_f зависит не только от f , но и от той машины Тьюринга, которую мы выбрали для вычисления f . Так что, когда мы пишем $a_f(x) = g(x)$, мы подразумеваем, что существует машина Тьюринга, вычислившая f , для которой размер ленты, использованной при вычислении $f(x)$, есть $g(x)$.

(с) Для явных преобразований¹⁾ получаем, что если g вычислима и $f(x_1, \dots, x_n) = g(\xi_1, \dots, \xi_k)$, то

$$a_f(x_1, \dots, x_n) = a_g(\xi_1, \dots, \xi_k) + l(x_1, \dots, x_n).$$

Доказательство. Мы приведем только набросок доказательства пункта (b); остальные пункты доказываются аналогично. В силу рекурсивного определения $f(x, y)$ требуется вычислить $h(x, 0, g(x))$, $h(x, 1, h(x, 0, g(x)))$ и т. д. В каждом из этих вычислений значений функции h фигурируют аргументы x , поэтому удобно предположить, что при вычислении функции на машине Тьюринга начальные аргументы неизменно остаются на ленте. Так, мы считаем, что данные машины Z_g и Z_h работают таким образом, что в конце на ленте написаны и начальные входы и вычисленные значения функций. (Это соглашение принято, например, в [4, стр. 320].) Принимая это во внимание, мы можем построить машину Z_f , на ленте которой в начале работы написано (x, y) , а в конце — $(x, y, f(x, y))$. Мы приводим здесь последовательность лент машины на различных узловых этапах вычисления и предоставляем читателю детальное построение такой машины. (Приведенная ниже последовательность написана в предположении, что $y \geq 2$; если y есть 0 или 1, то вычисление завершится быстрее.)

*

x	y	y	x	$g(x)$

 Переписывается y , переписывается x , вычисляется $g(x)$.

*

x	y	y	x	$g(x)$	0	$g(x)$

 Печатается 0, переписывается $g(x)$.

x	y	y	x	0	$g(x)$

 Стирается самое левое $g(x)$, уничтожается пробел.

x	y	y	x	0	$g(x)$	$h(x, 0, g(x))$

 Так как y не 0, вычисляется $h(x, 0, g(x)) = f(x, 1)$.

x	y	y	x	1	$f(x, 1)$

 Вычисляется $s(0)$, стирается $g(x)$, уничтожается пробел.

¹⁾ Это понятие, введенное Смутьяном [8, стр. 21], широко используется на протяжении всей статьи. Для полноты приведем определение: говорят, что функция f определяется из функции g явным преобразованием, если f есть $\lambda(x_1, \dots, x_n)g(\xi_1, \dots, \xi_k)$ (или, что эквивалентно, $f(x_1, \dots, x_n) = g(\xi_1, \dots, \xi_k)$ для всех $(x_1, \dots, x_n)$), где для каждого $i = 1, 2, \dots, k$ либо $\xi_i = x_j$ для некоторого $j = 1, 2, \dots, n$, либо ξ_i есть константа.

$$\boxed{x \mid y \mid y-1 \mid x \mid 1 \mid f(x, 1) \mid h(x, 1, f(x, 1))}$$

Из y вычитается 1;
так как $y-1 \neq 0$,
вычисляется
 $h(x, 1, f(x, 1)) = f(x, 2)$.

•
•
•

$$* \boxed{x \mid y \mid y-k \mid x \mid k \mid f(x, k) \mid h(x, k, f(x, k))} \quad \text{и т. д.}$$

•
•
•

$$\boxed{x \mid y \mid 1 \mid x \mid y-1 \mid f(x, y-1) \mid h(x, y-1, f(x, y-1))}$$

$$\{h(x, y-1, f(x, y-1)) = f(x, y)\}.$$

$$\boxed{x \mid y \mid 1 \mid x \mid y \mid f(x, y)}$$

Вычисляется $s(y-1)$, стирается
 $f(x, y-1)$, уничтожается пробел.

$$\boxed{x \mid y \mid f(x, y)}$$

Из 1 вычитается 1, в результате полу-
чается 0, стирается 0, x , y , уничтожается
пробел.

Пусть в приведенном выше вычислении k есть целое между 0 и y , такое, что $a_h(x, k, f(x, k)) = \max_{0 \leq z \leq y-1} a_h(x, z, f(x, z))$. Наи-

больший размер использованной ленты достигается при получении одной из трех лент, отмеченных выше звездочками. Если это будет при получении первой ленты, то он равен $l(x, y) + l(y) + a_g(x) + 2$, где 2 — количество пробелов, разделяющих числа на ленте и не учтенных еще в $l(x, y)$. Если это будет при получении второй ленты, то он равен $2[l(x, y) + l(g(x))] + 3$, и, наконец, в последнем случае, он равен $l(x, y) + l(y-k) + a_h(x, k, f(x, k)) + 2$. Заметив, что $a_F(z) \geq l(z) + l(F(z)) + 1$ для любой функции $F(z)$, мы можем заменить каждую из двух первых оценок на верхнюю границу $2[l(x, y) + a_g(x)]$, а третью оценку на $2l(x, y) + a_h(x, k, f(x, k))$. Таким образом, показано, что во всех случаях размер использованной ленты меньше или равен границе, данной в пункте (b) теоремы.

Полученные в теореме 1 оценки будут использованы в дальнейшем для классификации числовых функций. Как было уже замечено, функции из F являются простыми: их вычисление не только очень быстро, но также использует память (ленту), необходимую лишь для того, чтобы записать выход (на фиксированное число ячеек больше, чем нужно для записи входа). В приложении 2 мы приводим необходимые результаты о конечных автоматах и классе F . Читатель, который не знаком с ко-

нечными автоматами, может справиться с ними в этом приложении.

Таким образом, мы выбираем класс F функций, вычислимых (в двоичной системе счисления) конечными автоматами, в качестве исходного класса и используем эти функции, чтобы получить более широкие классы функций, для вычисления которых требуется больше времени и ленты. Рассматриваем класс F_1 всех числовых функций f , вычислимых на машинах Тьюринга, для которых функция a_f , дающая размер ленты, использованной при вычислении f , ограничена функцией из F , т. е. F_1 — это класс всех функций f , таких, что a_f ограничена функцией, вычислимой некоторым конечным автоматом.

В этом месте нужно сделать два замечания. Во-первых, мы предполагаем, что наши функции кодируются на машинах Тьюринга в двоичной системе счисления, а не в унарной, которая обычно используется в теории вычислимости. Но это соглашение гораздо ближе к реальной вычислительной практике и потому дает более естественные классы F_i ; оно выгодно еще и потому, что эти классы получаются более широкими, чем если бы мы использовали унарную систему счисления¹⁾. Второе замечание заключается в том, что мы требуем только, чтобы a_f была ограничена функцией из F , но не требуем, чтобы сама a_f принадлежала F . Это не только упрощает наши последующие доказательства²⁾, но также оказывается ближе к реальной вычислительной практике. На практике получают верхнюю границу для объема памяти, необходимой при вычислении, причем эта граница не обязательно достигается.

Учитывая эти замечания, читатель может легко убедиться в том, что умножение и функция a^x для любой фиксированной константы a принадлежат F_1 . Функция 2^{2^x} не принадлежит F_1 , так как длина 2^{2^x} есть $2^x + 1$ (и, следовательно, число ячеек использованной ленты не меньше этой величины). Но функция $2^x + 1$ не принадлежит F , так как она не удовлетворяет послед-

¹⁾ Например, в унарной системе счисления длина произведения двух чисел равна этому произведению, тогда как в двоичной системе эта длина не превосходит суммы длин чисел. Значит, умножение принадлежит F_1 , так как мы представляем числа в двоичной системе. Легко видеть, что оно не лежало бы в F_1 , если бы мы кодировали в унарной системе. (В действительности любая n -ичная система счисления ($n > 1$) дает в точности те же самые классы F_i , в чем читатель может легко убедиться.)

²⁾ Действительно, это не ограничивает общности. Если нам дана машина Тьюринга Z , вычисляющая f , для которой a_f ограничена $g \in F$, то легко построить машину Z' , вычисляющую f , для которой $a_{f'} = 2 \cdot g$. Машина Z' сначала переписывает аргументы x , вычисляет $g(x)$ и печатает что-нибудь в $2 \cdot g(x)$ ячейках, затем стирает все, кроме аргументов, и наконец имитирует Z . Подобные замечания имеют место для всех классов F_i .

ней теореме приложения 2. Так как 2^x принадлежит F_1 , то длина 2^{2^x} (а также, как легко можно показать, и размер ленты, необходимой для вычисления) лежит в F_1 . Теперь определяем класс F_2 всех функций f , таких, что a_f лежит в F_1 , и т. д.

Точнее, мы дадим следующее определение.

Определение 1. Числовая функция f принадлежит классу F_1 тогда и только тогда, когда

(I) она вычислима и

(II) существует функция $g \in F$, такая, что $a_f(x_1, \dots, x_n) \leq g(x_1, \dots, x_n)$ для всех n -ок $(x_1, \dots, x_n)$ неотрицательных целых чисел (короче, a_f ограничена функцией из F).

Для каждого $i > 1$ F_i есть класс всех вычислимых функций f , таких, что a_f ограничена функцией из F_{i-1} . Мы называем объединение всех классов F_i классом **предсказуемо вычислимых функций** и обозначаем его через $\mathcal{F}$.

Очевидно, что F_i строго включен в F_{i+1} ($i \geq 1$), так как $2^x \in F_1$, 2^{2^x} лежит в F_2 , но не лежит в F_1 , $2^{2^{2^x}}$ принадлежит F_3 , но не принадлежит F_2 и т. д.

В следующих двух разделах мы исследуем свойства классов F_i и показываем, что их объединение $\mathcal{F}$ совпадает с классом элементарных функций.

2. СВОЙСТВА КЛАССОВ F_i

В этом разделе мы будем устанавливать свойства замкнутости классов F_i относительно явных преобразований, композиции и рекурсивных определений. Для явных преобразований имеем следующий простой результат.

Лемма 1. *Каждый класс F_i замкнут относительно явных преобразований.*

Доказательство. Если $f(x_1, \dots, x_n) = g(\xi_1, \dots, \xi_m)$ для $g \in F_i$, тогда $a_f(x_1, \dots, x_n) = a_g(\xi_1, \dots, \xi_m) + l(x_1, \dots, x_n)$, где $a_g \in F_{i-1}$. Но функция a'_g , определенная равенством

$$a'_g(x_1, \dots, x_n) = a_g(\xi_1, \dots, \xi_m),$$

есть явное преобразование функции a_g и $a_g \in F_{i-1}$, если $i > 1$, или $a_g \in F$, если $i = 1$. Таким образом, по индукции, так как F замкнут относительно явных преобразований, получаем, что этим свойством обладает и каждый класс F_i .

Что касается композиции, то лемма 2, сформулированная ниже, показывает, где лежит композиция функций, пока же мы

отметим, что результат композиции может лежать в более высоком классе иерархии. Например,

$$2^x \in F_1, \quad 2^{2^x} \in F_2, \quad 2^{2^{2^x}} \in F_3$$

и т. д.

Лемма 2. Для любых положительных целых i и j если $h \in F_i$ и $g \in F_j$, то $f \in F_{i+j}$, где $f(x, y) = h(x, g(y))$.

Доказательство. Докажем сначала, что если g и g' лежат в F_j , то функция $f'(x, y) = g(x) + g'(y)$ тоже лежит в F_j . Заметим, что размер ленты, необходимой для сложения двух чисел, равен числу ячеек, необходимых для их записи: мы можем писать сумму прямо над аргументами. Таким образом, $a_{f'}(x, y) = a_g(x) + a_{g'}(y)$, так как, чтобы вычислить $f'(x, y)$, мы можем сначала вычислить $g(x)$ и $g'(y)$, а затем сложить два результата. Отсюда, если F_{j-1} замкнут относительно сложения, то и F_j тоже замкнут. Так как F замкнут относительно сложения (см. приложение 2), мы получаем наш результат индукцией. Далее, если $g \in F_j$ и $h \in F$, то $f'(x, y) = h(x, g(y))$ лежит в F_j , так как по приложению 2 $a_h(x, g(y)) \leq l(x) + l(g(y)) + K$. Но это не превосходит суммы $l(x) + a_g(y) + K$, которая лежит в F_{j-1} , значит, $f' \in F_j$, так как, чтобы вычислить $f'(x, y)$, мы можем сначала вычислить $g(y)$, а затем вычислить h , примененную к x и $g(y)$.

Вернемся теперь к доказательству самой леммы. Рассмотрим сначала случай, когда $i = 1$, а j произвольно. По теореме 1(b) имеем $a_f(x, y) \leq a_h(x, g(y)) + a_g(y) + l(x)$, где $a_h \in F$ и $a_g \in F_{j-1}$. Из предыдущих рассуждений получаем, что первый член неравенства лежит в F_j , так что $f \in F_{j+1}$, как и следовало ожидать. Теперь предположим, что лемма верна для $i = k$, j произвольно и что $h \in F_{k+1}$. Тогда имеем $a_h \in F_k$, откуда по индуктивному предположению $a_h(x, g(y)) \in F_{k+j}$ и, следовательно, $f(x, y) \in F_{(k+1)+j}$, что и завершает доказательство.

Чтобы иметь обозначения для функций 2^x , 2^{2^x} и т. д., введем следующее определение.

Определение 2. Пусть $f_0(x) = x$ и $f_{i+1}(x) = 2^{f_i(x)}$ для всех $i \geq 1$.

Покажем теперь, что не только $f_i \in F_i$, но и каждая функция из F_i ограничена f_i в следующем смысле. Если $f \in F_i$, то для некоторой константы K $f(x) < f_i(K \max(x, 1))$, где $\max(x, 1)$ есть наибольшее из чисел $x_1, \dots, x_n, 1$.

Лемма 3. Для любого $i > 1$ и любой $f \in F_i$ существует целое K , такое, что $f(x) < f_i(K \max(x, 1))$ и $a_f(x) < f_{i-1}(K \max(x, 1))$.

Доказательство. Если $f \in F_1$, то $a_f(x) < K \max(x, 1)$ по последней теореме приложения 2. Следовательно,

$$f(x) < 2^{l[f(x)]} < 2^{a_f(x)} < 2^{K \max(x, 1)} = f_1(K \max(x, 1)).$$

Предположим теперь, что лемма верна для F_i , и докажем ее для F_{i+1} . Так как $f \in F_{i+1}$, то $a_f(x) < f_i(K \max(x, 1))$. Тогда имеем

$$\begin{aligned} f(x) &< f_1(l[f(x)]) < f_1(a_f(x)) < f_1(f_i(K \max(x, 1))) = \\ &= f_{i+1}(K \max(x, 1)), \end{aligned}$$

что и завершает доказательство этой леммы.

Теперь мы можем доказать лемму 4. Она утверждает, что ограниченная рекурсия¹⁾ не выводит из $\mathcal{F}$. Фактически она указывает конкретный класс F_i , в котором будет лежать функция, определенная ограниченной рекурсией.

Лемма 4. Для любых положительных целых i и k если $h \in F_i$, $j \in F_k$ и $g \in F_{i+k}$, тогда $f \in F_{i+k}$, где f определяется из g , h и j посредством ограниченной рекурсии. (Следуя Гжегорчику, говорим, что f определяется из g , h и j при помощи ограниченной рекурсии, если $f(x, 0) = g(x)$, $f(x, s(y)) = h(x, y, f(x, y))$ и $f(x, y) \leq j(x, y)$; т. е. f определяется примитивной рекурсией из g и h и f ограничена j .)

Доказательство. По теореме 1(b)

$$a_f(x, y) \leq 2 \left[\max_{z < y} \left(\max a_h(x, z, f(x, z)), a_g(x) \right) + l(x, y) \right].$$

По лемме 3 и условию, что $f(x, z) \leq j(x, z)$, имеем

$$\begin{aligned} a_f(x, y) &\leq \\ &\leq 2 \left[\max_{z < y} \left[\max f_{i-1}(K \max(x, z, f_k(K' \max(x, z)))) \right], a_g(x) \right] + l(x, y). \end{aligned}$$

Отсюда мы получаем

$$\begin{aligned} a_f(x, y) &\leq \\ &\leq 2 \left[\max [f_{i-1}(K \max(x, y, f_k'(K' \max(x, y)))), a_g(x)] + l(x, y) \right] \leq \\ &\leq 2 [f_{i-1}(K f_k(K' \max(x, y))) + a_g(x) + l(x, y)]. \end{aligned}$$

¹⁾ Конечно, мы не можем надеяться на то, что объединение $\bigcup_{i=1}^{\infty} F_i$ за-

мкнуто относительно примитивной рекурсии; для вычисления функции, определенной при помощи примитивной рекурсии, по нашей схеме потребуется лента с длиной, зависящей от самой f , так как появится член $a_h(x, z, f(x, z))$. Но так как мы имеем границу для f , то можем ограничить и a_f . Такая граница для функции, определенной при помощи примитивной рекурсии, была введена еще Гжегорчиком [3] в понятие ограниченной рекурсии.

Покажем теперь, что мы ограничили a_f функцией из $F_{(i+k)-1}$. В-первых, так как $f_1(K' \max(x, y))$, очевидно, лежит в F_1 , то по лемме 2 $f_k(K' \max(x, y))$ принадлежит F_k . Так как F_k замкнут относительно сложения, то $K \cdot f_k(K' \max(x, y))$ лежит в F_k ; далее, по лемме 2 композиция f_{i-1} с этой функцией лежит в $F_{(i-1)+k}$. Поскольку $F_{(i-1)+k} = F_{(i+k)-1}$ замкнут относительно сложения, мы получаем желаемый результат, откуда следует, что $f \in F_{i+k}$.

Объединяя наши результаты, получаем следующую теорему.

Теорема 2. *Класс $\mathcal{E}$ элементарных функций содержится в объединении $\mathcal{F}$ последовательности классов F_i .*

Доказательство. Гжегорчик показал [3, теорема 2.5], что $\mathcal{E}$ — это наименьший класс функций, содержащий функцию следования и экспоненциальную функцию x^y , который замкнут относительно явных преобразований, композиции и ограниченной рекурсии¹⁾. Как уже было замечено, каждый класс F_i содержит

функцию следования; по леммам 1, 2 и 4 $\bigcup_{i=1}^{\infty} F_i$ замкнуто отно-

сительно трех необходимых операций. Чтобы доказать теорему, нам осталось только показать, что экспоненциальная функция

принадлежит $\bigcup_{i=1}^{\infty} F_i$. Действительно, x^y лежит в F_2 (применяем

один раз ограниченную рекурсию к функции $x \cdot y$, взяв в качестве границы $2^{2^{x+y}}$ — мы опускаем непосредственные выкладки).

3. РАВЕНСТВО $\mathcal{F}$ И $\mathcal{E}$

Мы только что показали, что $\mathcal{E} \subseteq \bigcup_{i=1}^{\infty} F_i = \mathcal{F}$. Докажем теперь обратное включение. Для этого мы проведем арифметизацию классов F_i в $\mathcal{E}$.

Выполненная в [1, стр. 56—62] арифметизация машин Тьюринга приводит к элементарному отношению T_n для каждого положительного целого n и к элементарной функции²⁾ U . Мы должны модифицировать эту арифметизацию, так как в [1] числовые функции кодируются наборами 1 и любой выход рассмат-

¹⁾ Гжегорчик не использовал понятия явных преобразований, однако легко видеть, что это просто подходящая перефразировка его теоремы 2.5.

²⁾ Дэвис замечает в [1] только то, что T_n и U примитивно рекурсивны. На самом деле они являются элементарными, поскольку все минимизации и квантификации ограничены. Об этих свойствах замкнутости элементарных функций и отношений см. [3]. (Мы можем определить элементарные отношения как такие отношения, характеристические функции которых элементарны. Это, очевидно, эквивалентно определению, данному в [3].)

ривается как изображение числа n , если лента содержит n единиц, распределенных на ней любым способом. Наше же соглашение кодировать числа цепочками из 0 и 1 (т. е. в двоичной системе счисления) требует, чтобы выходная лента в точности состояла из двоичного представления числа: только в этом случае она рассматривается как числовой выход.

К счастью, нужное нам видоизменение этой арифметизации оказывается очень простым. Мы должны изменить только функции MR и $Cogn$ и отношение Fin . Получающаяся в результате арифметизация дает элементарные U и T_n , такие, что n -местная функция f вычислима (в двоичной системе счисления) тогда и только тогда, когда существует число z_0 , такое, что для всех $x_1, \dots, x_n$

$$f(\mathbf{x}) = U(\min y [T_n(z_0, \mathbf{x}, y)]).$$

Чтобы выполнить это, переопределяем MR так, чтобы $MR(n)$ было гёделевым номером ленты, которая состоит в точности из двоичного представления n ; переопределяем $Cogn$ так, чтобы $Cogn(x) = n$ тогда и только тогда, когда $MR(n) = x$, и, наконец, переопределяем Fin так, чтобы $Fin(x, z)$ было верно тогда и только тогда, когда $Fin(x, z)$ верно в смысле Дэвиса, а также лента, входящая в описание с номером x , есть двоичное представление некоторого числа. Элементарные определения MR , $Cogn$ и Fin даны в приложении 1.

Предположим, что n -местная числовая функция f принадлежит F_i . Тогда, в частности, она вычислима, так что существует z_0 , для которого

$$f(\mathbf{x}) = U(\min y [T_n(z_0, \mathbf{x}, y)]).$$

Для того чтобы показать, что функция f элементарна, достаточно показать, что y , рассматриваемое как функция от $\mathbf{x}$, ограничено некоторой элементарной функцией g . Ибо тогда

$$f(\mathbf{x}) = U(\min y \leq g(\mathbf{x}) [T_n(z_0, \mathbf{x}, y)]),$$

где правая часть элементарна.

Чтобы показать, что существует элементарная функция g , которая ограничивает y , рассуждаем следующим образом. Максимальное число ячеек ленты, которые могут быть использованы на каком-нибудь шаге вычисления на данной машине Тьюринга Z , есть $a_f(\mathbf{x})$. По лемме 3 это меньше, чем $h(\mathbf{x})$, где

$$h(\mathbf{x}) = f_{i-1}(K \max(\mathbf{x}, 1)),$$

и согласно [3] функция h элементарна. Машина Z имеет конечное число m символов, которые могут появляться на ленте. Таким образом, число различных лент, которые могут возникнуть при вычислении, меньше, чем $m^{h(\mathbf{x})}$, кроме того, у Z

конечное число внутренних состояний, например k . Следовательно, общее число мгновенных описаний Z (а значит, и число шагов¹⁾), которые могут появиться в данном вычислении, меньше, чем $j(\mathbf{x}) = k \cdot h(\mathbf{x}) \cdot m^{h(\mathbf{x})}$, где $j(\mathbf{x})$ является элементарной функцией $\mathbf{x}$. Предположим теперь, что $q(\mathbf{x})$ — максимальный из гёделевых номеров мгновенных описаний, возникающих в этом вычислении. Тогда граница $g(\mathbf{x})$ равна²⁾ $[\text{Pr}(j(\mathbf{x}))^{q(\mathbf{x})}]^{j(\mathbf{x})}$, так как она превышает число всех последовательностей длины $j(\mathbf{x})$, гёделев номер каждого члена которых меньше $q(\mathbf{x})$. Следовательно, функция g элементарна, если функция q такова же. Чтобы показать, что q элементарна, допустим, что M — максимальный из гёделевых номеров всех внутренних состояний Z и символов, которые могут появляться на ленте Z . Тогда мы можем положить

$$q(\mathbf{x}) = [\text{Pr}(h(\mathbf{x}) + 1)^M]^{h(\mathbf{x}) + 1}.$$

Следовательно, g элементарна. Как мы уже говорили, отсюда следует, что любая функция f из F_i элементарна.

Таким образом, мы только что показали, что $\mathcal{F} = \bigcup_{i=1}^{\infty} F_i \subseteq \mathcal{E}$.

Вместе с теоремой 2 это дает нам следующий основной результат.

Теорема 3. *Класс $\mathcal{F}$ предсказуемо вычислимых функций есть в точности класс элементарных функций, т. е. $\mathcal{F} = \mathcal{E}$.*

4. КЛАССЫ F' И $\mathcal{E}^2$

Класс $\mathcal{F}$ предсказуемо вычислимых функций определяется в терминах сложности вычислений на машинах Тьюринга. Однако, доказав, что этот класс совпадает с классом $\mathcal{E}$ элементарных функций, мы охарактеризовали $\mathcal{F}$ в теоретико-числовых терминах, не ссылаясь на машины Тьюринга. В этом разделе мы закладываем основу для подобной характеристики каждого индивидуального класса F_i .

В предыдущем разделе мы получили некоторые сведения о порядке роста и свойствах замкнутости классов F_i . Ясно, однако, что полученных результатов не достаточно для того, чтобы охарактеризовать эти индивидуальные классы. Рассмотрим, напри-

¹⁾ Если бы число шагов в вычислении было больше $j(\mathbf{x})$, то Z дважды достигла бы одного и того же мгновенного описания и поэтому стала бы повторять цикл мгновенных описаний и никогда бы не остановилась.

²⁾ Следуя [1], мы обозначаем n -е простое число через $\text{Pr}(n)$, полагая $\text{Pr}(0) = 0$.

мер, функцию h из F_1 , где $h(x, y) = 2^x \cdot y$. Тогда лемма 2 гарантирует нам, что результат подстановки f_1 вместо какого-нибудь аргумента h есть функция из F_2 ; в действительности же мы можем легко показать, что $f \in F_1$, где $f(x, y) = 2^x \cdot 2^y = h(x, 2^y)$. Конечно, это выполняется потому, что h зависит от своего второго аргумента скорее «мультипликативно», чем «экспоненциально», точнее, потому, что a_h имеет вид $a_h(x, y) = q(x) + Kl(y)$ для некоторых $q \in F$, $K \geq 0$. Вообще имеем следующий результат.

Лемма 5. *Предположим, что функция $h \in F_i$ обладает тем свойством, что существует $q \in F_{i-1}$, такая, что $a_h(x, y) \leq q(x) + Kl(y)$. Тогда функция f , определенная посредством $f(x, z) = h(x, g(z))$, принадлежит F_i , если только g принадлежит F_i . Далее, если g, h и j принадлежат F_i и $a_h(x, y, z) \leq q(x, y) + Kl(z)$ с $q \in F_{i-1}$, то функция f , определенная из них ограниченной рекурсией, принадлежит F_i .*

Доказательство.

$$\begin{aligned} a_f(x, z) &\leq a_h(x, g(z)) + a_g(z) + l(x) \leq \\ (a) \quad &\leq q(x) + Kl(g(z)) + a_g(z) + l(x) \leq \\ &\leq q(x) + (K+1)a_g(z) + l(x), \end{aligned}$$

где $q, a_g \in F_{i-1}$ и поэтому $f \in F_i$.

$$\begin{aligned} a_f(x, y) &\leq 2 \left[\max_{z < y} [a_h(x, z, f(x, z)), a_g(x)] + l(x, y) \right] \leq \\ &\leq 2 \left[\max_{z < y} [q(x, z) + Kl(f(x, z)), a_g(x)] + l(x, y) \right] \leq \\ (b) \quad &\leq 2 \left[\max_{z < y} [q(x, z) + Kl(j(x, z)), a_g(x)] + l(x, y) \right] \leq \\ &\leq 2 \left[\max_{z < y} [q(x, z) + Ka_j(x, z), a_g(x)] + l(x, y) \right]. \end{aligned}$$

Здесь мы воспользуемся леммой 3, чтобы получить монотонные функции, и, продолжая, получим

$$\begin{aligned} a_f(x, y) &\leq 2 \left[\max_{z < y} [f_{i-1}(K_1 \max(x, z)) + \right. \\ &\quad \left. + Kf_{i-1}(K_2 \max(x, z)), a_g(x)] + l(x, y) \right] \leq \\ &\leq 2 \left[\max [f_{i-1}(K_1 \max(x, y)) + \right. \\ &\quad \left. + Kf_{i-1}(K_2 \max(x, y)), a_g(x)] + l(x, y) \right]. \end{aligned}$$

Так как $\max(F, G) \leq F + G$, мы получаем $f \in F_i$, что и требовалось показать.

Наше описание каждого F_i будет существенно использовать функции с «мультипликативным» свойством, встречающимся в посылках леммы 5. Сейчас мы сконцентрируем наше внимание на особенно полезном подклассе F' класса F_1 (который, как мы покажем, есть в точности класс $\mathcal{E}^{21}$), все элементы которого обладают этим «мультипликативным» свойством.

Определение 3. F' есть класс всех вычислимых функций f , для которых размер ленты, используемой в вычислении $f(x_1, \dots, x_n)$, не превосходит произведения некоторой константы K_f на $l(x_1, \dots, x_n)$. Т. е. для всех $x_1, \dots, x_n$

$$\begin{aligned} a_f(x_1, \dots, x_n) &\leq l(x_1\beta \dots \beta x_n x_1\beta \dots \beta x_n \dots x_1\beta \dots \beta x_n) = \\ &= K_f(l(x_1) + l(x_2) + \dots + l(x_n) + (n-1)). \end{aligned}$$

Приступим к получению различных свойств замкнутости F' .

Лемма 6. Класс функций F' замкнут относительно явных преобразований, композиции и ограниченной рекурсии.

Доказательство. (а) Пусть $g \in F'$, $a_g(x_1, \dots, x_m) \leq K_g l(x_1, \dots, x_m)$ и $f(x_1, \dots, x_n) = g(\xi_1, \dots, \xi_m)$. Тогда по теореме 1

$$a_f(x_1, \dots, x_n) \leq K_g l(\xi_1, \dots, \xi_m) + l(x_1, \dots, x_n),$$

и так как $l(\xi_1, \dots, \xi_m) \leq Cl(x_1, \dots, x_n)$ (для некоторой константы C , зависящей от того, какие ξ_i являются константами, а какие — переменными), мы имеем, что

$$a_f(x_1, \dots, x_n) \leq (K_g C + 1) l(x_1, \dots, x_n).$$

Таким образом, в качестве K_f можно взять $K_g C + 1$.

(б) Пусть $g, h \in F'$, $a_g \leq K_g l$, $a_h \leq K_h l$ и f определяется посредством $f(\mathbf{x}, \mathbf{y}) = h(\mathbf{x}, g(\mathbf{y}))$. По теореме 1 мы имеем (используя, что $l(\mathbf{x}, \mathbf{y}) = l(\mathbf{x}) + l(\mathbf{y}) + 1$ и $lg(\mathbf{y}) \leq a_g(\mathbf{y}) \leq K_g l(\mathbf{y})$)

$$\begin{aligned} a_f(\mathbf{x}, \mathbf{y}) &\leq [l(\mathbf{x}) + l(\mathbf{y}) + 1] + \max[l(\mathbf{x}) + K_g l(\mathbf{y}), K_h(l(\mathbf{x}) + \\ &\quad + lg(\mathbf{y}) + 1)] \leq [l(\mathbf{x}) + l(\mathbf{y}) + 1] + K_h[l(\mathbf{x}) + K_g l(\mathbf{y}) + 1] \leq \\ &\leq [(K_h + 1)K_g] l(\mathbf{x}, \mathbf{y}). \end{aligned}$$

Итак, K_f можно взять равным $(K_h + 1)K_g$.

¹⁾ Класс $\mathcal{E}^2$ был впервые введен в [3]. Мы повторяем его определение в теореме 4, используя понятие явных преобразований.

(с) Пусть $g, h, j \in F'$ и K_g, K_h, K_j — связанные с ними константы. Пусть f определяется из g, h, j при помощи ограниченной рекурсии. Тогда по теореме 1

$$\begin{aligned} a_f(x, y) &\leq 2 \left[\max \left[\max_{z < y} (K_h l(x, z, f(x, z))), K_g l(x) \right] + l(x, y) \right] \leq \\ &\leq 2 \left[\max \left[\max_{z < y} (K_h l(x, z) + K_h K_j l(x, z)) + K_h, K_g l(x) \right] + l(x, y) \right] \leq \\ &\leq 2 \left[\max [K_h l(x, y) + K_h K_j l(x, y) + K_h, K_g l(x)] + l(x, y) \right] \leq \\ &\leq 2 [\max (K_h (K_j + 2), K_g) + 1] \cdot l(x, y). \end{aligned}$$

Таким образом, мы можем положить K_f равным $2[\max (K_h (K_j + 2), K_g) + 1]$.

В качестве примера использования этой леммы мы можем показать, что сложение и умножение суть функции из F' . Очевидно, что функция следования вычисляется на машине Тьюринга, использующей ровно на одну ячейку больше, чем необходимо для записи входа. Далее, сочленение¹⁾ тоже принадлежит F' , а сложение определяется из функции следования посредством рекурсии, ограниченной сочленением. В свою очередь умножение определяется из сложения посредством рекурсии, ограниченной сочленением, так что сложение и умножение принадлежат F' .

Для того чтобы объединить наши сведения о классе F' , формулируем следующую теорему (это просто краткое изложение леммы 6 и предыдущего абзаца):

Теорема 4. F' содержит класс $\mathcal{E}^2$ (где $\mathcal{E}^2$ есть наименьший класс, содержащий функции следования и умножения и замкнутый относительно явных преобразований, композиции и ограниченной рекурсии).

Охарактеризуем теперь класс F' без ссылок на машины Тьюринга или ограниченные вычисления. Мы только что показали, что $\mathcal{E}^2$ является подклассом F' . Оставшаяся часть этого раздела посвящается арифметизации F' в пределах $\mathcal{E}^2$ и, следовательно, установлению равенства этих двух классов. Так как определение $\mathcal{E}^2$ является чисто теоретико-числовым, то это дает нам желаемое описание F' .

¹⁾ Сочленение (двоичное) есть функция, которая, будучи применена к числам с двоичными представлениями $a_n a_{n-1} \dots a_1 a_0$ и $b_m b_{m-1} \dots b_1 b_0$, дает число, двоичное представление которого есть $a_n a_{n-1} \dots a_1 a_0 b_m b_{m-1} \dots b_1 b_0$. Например, сочленение 10 и 11 есть 1011, а сочленение 0 и 11 есть 011 — число 3.

Мы начнем с рассмотрения границы вычисления функции f , налагая на нее требование принадлежать F' . Чтобы фиксировать обозначения в этом обсуждении, предположим, что машина, вычисляющая f , имеет гёделев номер z , обладает k внутренними состояниями (конфигурациями) и использует на ленте три символа¹⁾. Если аргументы суть $x_1, \dots, x_n$, то длина наибольшей ленты, которая может встретиться, есть $K \cdot l(x_1, \dots, x_n)$ (по определению F'). Поскольку каждая ячейка ленты может содержать один из трех различных символов, число возможных лент, которые могут встретиться в вычислении $f(x_1, \dots, x_n)$, не больше $3^{K \cdot l(x_1, \dots, x_n)}$. Таким образом, число возможных мгновенных описаний меньше, чем $3^{K \cdot l(x_1, \dots, x_n)} \cdot k \cdot l(x_1, \dots, x_n)$. Однако мы можем заменить эту границу на

$$[(x_1 + 1) \dots (x_n + 1)]^{2 \cdot K} \cdot 2^{2 \cdot K \cdot (2n-1)} \cdot k \cdot [x_1 + \dots + x_n + (n-1)],$$

которую мы будем обозначать через $b_{n,z}(x_1, \dots, x_n)$, ибо

$$\begin{aligned} 3^{K \cdot l(x_1, \dots, x_n)} &< [2^{l(x_1, \dots, x_n)}]^{2 \cdot K} = \\ &= [2^{l(x_1)} \dots 2^{l(x_n)}]^{2 \cdot K} \cdot [2^{n-1}]^{2 \cdot K} < \\ &< [2^n \cdot (x_1 + 1) \dots (x_n + 1)]^{2 \cdot K} \cdot 2^{2 \cdot K \cdot (n-1)}. \end{aligned}$$

Так как $b_{n,z}$ получена с использованием только сложения и умножения, то функция $b_{n,z}(x_1, \dots, x_n)$ принадлежит, очевидно, $\mathcal{E}^2$.

Чтобы использовать включение $b_{n,z}$ в $\mathcal{E}^2$, мы следующим образом переделаем нашу арифметизацию. Мы заменим обычное представление $f(x_1, \dots, x_n)$

$$U(\min y [T_n(z, x_1, \dots, x_n, y)])$$

для этого фиксированного значения z ²⁾ на

$$U'_{n,z}(x_1, \dots, x_n, \min y [T'_{n,z}(x_1, \dots, x_n, y)]).$$

Здесь отношение $T'_{n,z}$ имеет место для $x_1, \dots, x_n, y$ тогда и только тогда, когда y есть число мгновенных описаний вычисления, начинающегося с $(x_1, \dots, x_n)$, на машине с номером z . Далее, $U'_{n,z}$ есть число на ленте в y -м мгновенном описании этого вычисления (если на ленте представлено некоторое число,

¹⁾ В приложении 2 формулируется определение машины Тьюринга, работающей в алфавите из трех символов.

²⁾ Таким образом, мы утрачиваем «нормальную форму» в общепринятом смысле. Тем не менее, как показывает стандартная диагонализация аргумента, это необходимо (см. разд. 6).

0 — в противном случае). Так как по сказанному выше $T'_{n,z}(\mathbf{x}, y)$ означает, что $y < b_{n,z}(\mathbf{x})$, имеем

$$f(\mathbf{x}) = U'_{n,z}(\mathbf{x}, \min y \leq b_{n,z}(\mathbf{x}) | T'_{n,z}(\mathbf{x}, y)|).$$

Таким образом, как только мы сможем показать, что $U'_{n,z}$ и $T'_{n,z}$ лежат в $\mathcal{E}^2$ и связанном с ним классе $\mathcal{E}^2_*$ отношений¹⁾ соответственно, мы покажем, что f лежит в $\mathcal{E}^2$ и вообще $F' \subseteq \mathcal{E}^2$.

Чтобы дать точные описания $T'_{n,z}$ и $U'_{n,z}$, рассуждаем следующим образом. Предположим, что мы провели арифметизацию машин Тьюринга в пределах класса $\mathcal{E}^2$ функций и класса $\mathcal{E}^2_*$ отношений. Предположим, в частности, что мы определили функции Init_n , Yield и Decode в $\mathcal{E}^2$ и отношение Fin в $\mathcal{E}^2_*$ со следующей интерпретацией. $\text{Init}_n(x_1, \dots, x_n)$ есть гёделев номер мгновенного описания (некоторой) машины Тьюринга в ее начальном состоянии, когда она рассматривает в стандартном положении вход $(x_1, \dots, x_n)$ на ленте; $\text{Yield}(x, z)$ есть (гёделев) номер того мгновенного описания, которое непосредственно следует за мгновенным описанием с номером x на машине с номером z ; $\text{Decode}(x)$ есть число, изображенное лентой заключительного мгновенного описания, (гёделев) номер которого есть x (если эта лента является двоичной записью числа, в противном случае $\text{Decode}(x)$ есть 0); и, наконец, $\text{Fin}(x)$ верно тогда и только тогда, когда x есть (гёделев) номер заключительного мгновенного описания (в котором машина останавливается).

Читатель, знакомый с работой Гжегорчика [3] или Смутьяна [8] и обладающий небольшими познаниями в области перечислимых множеств, смог бы самостоятельно убедиться в том, что такую арифметизацию можно выполнить, хотя обычная арифметизация при помощи степеней простых чисел выводит¹⁾ из $\mathcal{E}^2$. Детально такая арифметизация проводится в приложении 3.

¹⁾ Класс $\mathcal{E}^2_*$ определяется и исследуется в [3]. Его можно определить как класс всех отношений, характеристические функции которых лежат в $\mathcal{E}^2$. (Мы уславливаемся обозначать через X_* класс отношений, характеристические функции которых лежат в классе X , для любых классов X функций.)

²⁾ Рассмотрим, например, Init_1 . Пусть гёделев номер символа a обозначается через $gn(a)$, пусть $x_1 = a_k a_{k-1} \dots a_1 a_0$, где каждый a_i есть либо 0, либо 1, и пусть m есть минимум из $gn(0)$ и $gn(1)$. Тогда

$$\text{Init}_1(x_1) = 2^{gn(a_k)} \cdot 3^{gn(a_{k-1})} \dots p_r(k)^{gn(a_1)} \cdot p_r(k+1)^{gn(a_0)} \geq [(k+1)!]^m.$$

Итак, $\text{Init}_1(x_1) \geq [(l(x_1) + 1)!]^m$ и, значит, не может лежать в $\mathcal{E}^2$; ее длина как функция от x_1 не ограничена линейной функцией, и поэтому она не лежит даже в F_1 .

Если даны функции Init_n , Yield , Decode и отношение Fin , мы можем построить функцию $U'_{n,z}$ и отношение $T'_{n,z}$ следующим образом. Для каждого фиксированного значения z определяем функцию Yield_z , полагая $\text{Yield}_z(x) = \text{Yield}(x, z)$. Так как $\mathcal{E}^2$ замкнут относительно явных преобразований, эта функция лежит в $\mathcal{E}^2$. Теперь при помощи ограниченной рекурсии, примененной к Yield_z , получаем $H_{n,z}$, значение которой от аргументов $(x_1, \dots, x_n, y)$ есть мгновенное описание y -го шага вычисления на машине с номером z , начинающегося с ленты, представляющей n -ку $(x_1, \dots, x_n)$;

$$\begin{aligned} H_{n,z}(\mathbf{x}, 0) &= \text{Init}_n(\mathbf{x}), \\ H_{n,z}(\mathbf{x}, s(y)) &= \text{Yield}_z(H_{n,z}(\mathbf{x}, y)), \\ H_{n,z}(\mathbf{x}, y) &\leq b_{n,z}(\mathbf{x}). \end{aligned}$$

Здесь ограничение на $H_{n,z}$ вытекает просто из того требования, что длина используемой ленты меньше, чем $K \cdot l(\mathbf{x})$. Отсюда следует, что наибольшее число, которое может встретиться на ленте, меньше $2^{K \cdot l(\mathbf{x})}$, что в свою очередь меньше, чем $b_{n,z}(\mathbf{x})$.

Теперь мы определим отношение $T'_{n,z}$, полагая

$$T'_{n,z}(\mathbf{x}, y) = \text{Fin}[H_{n,z}(\mathbf{x}, y)].$$

Поскольку $H_{n,z}$ принадлежит $\mathcal{E}^2$, мы получаем $T'_{n,z} \in \mathcal{E}_*^2$. Наконец, $U'_{n,z}(\mathbf{x}, y)$ равно такому числу, которое отвечает мгновенному описанию с номером $H'_{n,z}(\mathbf{x}, y)$. Таким образом, мы полагаем просто

$$U'_{n,z}(\mathbf{x}, y) = \text{Decode}[H_{n,z}(\mathbf{x}, y)]$$

и получаем, как и ожидалось, $U'_{n,z}(\mathbf{x}, y) \in \mathcal{E}^2$. Этот результат мы подытожим в теореме 5.

Теорема 5. *Функция f принадлежит F' тогда и только тогда, когда существует число z , такое, что для всех $x_1, \dots, x_n$*

$$f(\mathbf{x}) = U'_{n,z}(\mathbf{x}, \min y \leq b_{n,z}(\mathbf{x}) [T'_{n,z}(\mathbf{x}, y)]).$$

Доказательство. Если такое z существует, оно является номером машины Тьюринга, вычисляющей функцию из F' . Однако по определению $U'_{n,z}$ эта машина вычисляет функцию f . Обратно, если $f \in F'$, существует машина, например, с номером z , которая ее вычисляет, и опять по определению $U'_{n,z}$ и $T'_{n,z}$ теорема выполняется.

Следствие. $\mathcal{E}^2 = F'$,

Доказательство. По теореме 3 имеем $\mathcal{E}^2 \subseteq F'$. Теорема 5 показывает, что $F' \subseteq \mathcal{E}^2$, так как $U'_{n,z}$ лежит в $\mathcal{E}^2$ и $T'_{n,z}$ лежит в $\mathcal{E}^2_*$.

5. ОПИСАНИЕ КЛАССОВ F_i

В предыдущем разделе мы заявили, что, прежде чем охарактеризовать классы F_i , мы получим подклассы классов F_i , которые удовлетворяют условиям леммы 5 и, кроме того, содержат такие функции, как $2^x \cdot y$. Теперь, когда мы изучили класс F' , мы в состоянии это сделать. Именно, мы видим теперь, что сам F' удовлетворяет условиям леммы: любая функция из F' зависит «мультипликативно» от каждого аргумента. Зафиксируем номер некоторого аргумента (например, k -й) и расширим F' насколько это возможно посредством таких применений явных преобразований, композиции и ограниченной рекурсии, которые сохраняют эту «мультипликативную» зависимость от k -го аргумента. Таким образом, с каждым классом X , содержащим F' , мы связываем подкласс X^k (для любого $k \geq 1$), который состоит из функций, зависящих «мультипликативно» от своих k -х аргументов.

Определение 4. Если дан класс X функций, содержащий класс F' , определяем подкласс X^k класса X , требуя, чтобы функция $f \in X$ принадлежала X^k тогда и только тогда, когда выполняется одно из следующих условий:

- (a) $f \in F'$;
- (b) $f(x_1, \dots, x_n) = g(\xi_1, \dots, \xi_m)$ для $k \leq n$, $g \in X$ и далее $g \in X^l$ для каждого l такого, что $\xi_l = x_k$;
- (c) $f(x_1, \dots, x_l, y_1, \dots, y_m) = h(x_1, \dots, x_l, g(y_1, \dots, y_m))$, где либо (I) $k \leq l$ и $h \in X^k$, либо (II) $k = l + m$, $h \in X^{l+1}$ и $g \in X^m$;
- (d) $f(x_1, \dots, x_{k-1}, 0) = g(x_1, \dots, x_{k-1})$,
 $f(x_1, \dots, x_{k-1}, s(y)) = h(x_1, \dots, x_{k-1}, y, f(x_1, \dots, x_{k-1}, y))$,
 $f(x_1, \dots, x_{k-1}, y) \leq j(x_1, \dots, x_{k-1}, y)$,

где $g \in X$, $h \in X^k \cap X^{k+1}$ и $j \in X^k$.

Лемма 7. Если $f \in F_i^k$, то $a_f(x_1, \dots, x_n) \leq q(x_1, \dots, x_{k-1}, x_{k+1}, \dots, x_n) + K \cdot l(x_k)$ для некоторого целого $K \geq 0$ и некоторой $q \in F_{i-1}$.

Доказательство. $f \in F_i^k$ означает, что либо $f \in F'$, либо она образуется из функций F'_i при помощи правил (b), (c)

или (d). Если $f \in F'$, то лемма, очевидно, выполняется, так как

$$a_f(\mathbf{x}, y, z) \leq K \cdot [l(\max(\mathbf{x}, y, z))] \leq K \cdot [l(\mathbf{x}, y, z)] = \\ = K \cdot [l(\mathbf{x}, y) + 1] + K \cdot l(z).$$

Доказательство завершается непосредственно индукцией по числу применений (b), (c) и (d) в определении f .

Из лемм 5 и 7 следует, что если n -местная функция h лежит в F_i^n и g лежит в F_i , тогда функция f принадлежит F_i , где $f(x_1, \dots, x_{n-1}, y_1, \dots, y_m) = h(x_1, \dots, x_{n-1}, g(y_1, \dots, y_m))$. Далее, если $(n+1)$ -местная функция $h \in F_i^{n+1}$ и $g, j \in F_i$, тогда функция f также принадлежит F_i , где

$$f(x_1, \dots, x_{n-1}, 0) = g(x_1, \dots, x_{n-1}),$$

$$f(x_1, \dots, x_{n-1}, s(y)) = h(x_1, \dots, x_{n-1}, y, f(x_1, \dots, x_{n-1}, y))$$

и

$$f(x_1, \dots, x_{n-1}, y) \leq j(x_1, \dots, x_{n-1}, y).$$

Чтобы подытожить эти результаты, введем следующие два определения и затем сформулируем теорему 6.

Определение 5. Будем говорить, что класс X (функций) замкнут относительно **подстановок в функции класса** Y , если для любых $h \in Y$ и $g \in X$ будет $f \in X$, где $f(\mathbf{x}, y) = h(\mathbf{x}, g(y))$.

Определение 6. Будем говорить, что класс X замкнут относительно **ограниченной рекурсии по отношению к функциям из класса** Y , если для любых $h \in Y$, $g, j \in X$ будет $f \in X$, где

$$f(\mathbf{x}, 0) = g(\mathbf{x}),$$

$$f(\mathbf{x}, s(y)) = h(\mathbf{x}, y, f(\mathbf{x}, y)),$$

$$f(\mathbf{x}, y) \leq j(\mathbf{x}, y).$$

Теорема 6. Каждый класс F_i содержит класс $\mathcal{E}^2$ и функции $f_1 \circ f$ для любой $f \in F_{i-1}$. Далее, он замкнут относительно явных преобразований, подстановок в k -местные функции из F_i^k и ограниченной рекурсии по отношению к $(k+1)$ -местным функциям из F_i^{k+1} для всех $k \geq 1$.

Доказательство. Принадлежность $f_1 \circ f$ классу F_i следует из леммы 2. Замкнутость относительно явных преобразований есть результат леммы 1. Для доказательства остальных утверждений комбинируем леммы 5 и 7.

Следствие. F_i замкнут относительно подстановок в функции из $\mathcal{E}^2$ и ограниченной рекурсии по отношению к функциям $\mathcal{E}^2$.

Свойств классов F_i , описанных в теореме 6, достаточно, чтобы охарактеризовать эти классы. Мы сделаем это, определив сначала классы $\bar{F}_i$, которые, как мы потом докажем, в точности совпадают с классами F_i .

Определение 7. Для каждого положительного целого i класс $\bar{F}_i$ есть наименьший класс X функций, включающий $\mathcal{E}^2$ и содержащий функции $f_1 \circ f$ для любой $f \in \bar{F}_{i-1}$, если $i > 1$, или $f \in F$, если $i = 1$, который замкнут относительно явных преобразований, подстановок в k -местные функции из X^k и ограниченной рекурсии по отношению к функциям от $(k + 1)$ аргумента, лежащим в X^{k+1} , для всех $k \geq 1$. ($\bar{F}_i$ определены корректно, поскольку существуют классы X , удовлетворяющие этим условиям, и, так как $X^n \cap Y^n = (X \cap Y)^n$, пересечение таких классов снова удовлетворяет условиям.)

Определение 7 предназначено для того, чтобы отразить свойства классов F_i , перечисленные в теореме 6, так что мы немедленно получаем, что $\bar{F}_i \subseteq F_i$. Объединим непосредственные следствия из наших предыдущих результатов и этого определения.

Теорема 7. Для всех $i \geq 1$ $\bar{F}_i \subseteq F_i$, $\bar{F}_i \subset \bar{F}_{i+1}$, $\bar{F}_i \supset \mathcal{E}^2$ и $\bar{F}_i \subset \mathcal{E}$.

Доказательство. $\bar{F}_i \subseteq F_i$ по теореме 6, $\bar{F}_i \subset \bar{F}_{i+1}$ по определению, $\mathcal{E}^2 \subset \bar{F}_i$ снова по определению. То, что $\bar{F}_i \subset \mathcal{E}$, очевидно, так как умножение и 2^x лежат в $\mathcal{E}$.

Перейдем к доказательству того, что $\bar{F}_i = F_i$. Итак, каждая элементарная функция лежит в некотором $\bar{F}_i$. Заметим кстати, что этот результат дает новое описание класса $\mathcal{E}$ в терминах специального вида подстановок и рекурсий, появляющихся выше в определении 7.

Чтобы завершить описание классов F_i , мы хотим доказать, что $F_i \subseteq \bar{F}_i$. Для этого мы берем за образец доказательство того, что $F' \subseteq \mathcal{E}^2$, т. е. мы определяем $T_{n,z}^i \in (\bar{F}_i)_*$ и $U_{n,z}^i, b_{n,z}^i \in \bar{F}_i$ и показываем, что если $f \in F_i$, то

$$f(\mathbf{x}) = U_{n,z}^i(\mathbf{x}, \min y \leq b_{n,z}^i(\mathbf{x}) [T_{n,z}^i(\mathbf{x}, y)]).$$

Мы покажем в действительности, что

$$\min y \leq b_{n,z}^i(\mathbf{x}) [T_{n,z}^i(\mathbf{x}, y)]$$

определяет функцию из $\bar{F}_i$ и что $U_{n,z}^i \in \bar{F}_i^{n+1}$. Таким образом, будет показано, что $f(\mathbf{x})$ есть результат подстановки функции из $\bar{F}_i$ в $(n + 1)$ -местную функцию из $\bar{F}_i^{n+1}$ и, следовательно (по определению 7), $f(\mathbf{x})$ должна лежать в $\bar{F}_i$. Докажем сначала

теорему, необходимую для того, чтобы установить эти включения¹⁾.

Лемма 8. Для любой $f \in \bar{F}_i^{n+1}$ функция F , значение которой от $(x_1, \dots, x_n, y)$ равно количеству z , не превосходящих y и таких, что $f(x_1, \dots, x_n, z) = 0$, принадлежит $\bar{F}_i^{n+1}$.

Доказательство. Начнем с определения функций p , $\div$ и τ из $\mathcal{E}^2$.

$$\left. \begin{array}{l} p(0) = 0 \\ p(s(x)) = x \\ p(x) \leq x \end{array} \right\} p(x) \text{ есть число, предшествующее } x.$$

$$\left. \begin{array}{l} x \div 0 = x \\ x \div s(y) = p(x \div y) \\ x \div y \leq x \end{array} \right\} \begin{array}{l} x \div y \text{ есть } 0, \text{ если } x \leq y, \text{ и} \\ x - y \text{ в противном случае.} \end{array}$$

$$\left. \begin{array}{l} \tau(x, 0) = s(x) \\ \tau(x, s(y)) = x \\ \tau(x, y) \leq s(x) \end{array} \right\} \begin{array}{l} \tau(x, y) \text{ есть } s(x), \text{ если} \\ y = 0, \text{ и } x \text{ в противном случае.} \end{array}$$

Так как $f \in \bar{F}_i^{n+1}$, то и функция $t \in \bar{F}_i^{n+1}$, где

$$t(x_1, \dots, x_n, y, z) = \tau(z, f(x_1, \dots, x_n, s(y))).$$

Теперь определяем F при помощи ограниченной рекурсии по отношению к t , так что по определению 7 $F \in \bar{F}_i^{n+1}$:

$$F(x_1, \dots, x_n, 0) = 1 \div f(x_1, \dots, x_n, 0),$$

$$F(x_1, \dots, x_n, s(y)) = t(x_1, \dots, x_n, y, F(x_1, \dots, x_n, y)),$$

$$F(x_1, \dots, x_n, y) \leq s(y).$$

То, что F — искомая функция, следует из замечаний, что $1 \div z$ есть 1, если $z = 0$, и 0 в противном случае и что τ прибавляет 1 тогда и только тогда, когда ее второй аргумент есть 0.

Теорема 8. Если $f \in \bar{F}_i^{n+1}$, то функция g , определенная посредством $g(x_1, \dots, x_n, y) = \min z \leq y [f(x_1, \dots, x_n, z) = 0]$, также принадлежит $\bar{F}_i^{n+1}$.

Доказательство. По лемме 8 $F \in \bar{F}_i^{n+1}$. Применяя лемму 8 к самой F , мы получаем, что функция $G \in \bar{F}_i^{n+1}$, где $G(x, y)$ равна количеству z , меньших, чем $y + 1$, для которых $F(x, z) = 0$.

¹⁾ Мы следуем методу, использованному в [3] для доказательства того, что ограниченная минимизация допускается в широком множестве классов.

Но $F(x, z) = 0$ означает, что нет таких $w \leq z$, для которых $f(x, w) = 0$. Таким образом, $G(x, y)$ есть в точности количество z , меньших, чем $y + 1$, и таких, что для каждого $w \leq z$ $f(x, w) \neq 0$, т. е. $G(x, y)$ есть наименьшее $z \leq y$, такое, что $f(x, z) = 0$, если такое z существует, и $y + 1$, если такого z не существует. Чтобы изменить G так, чтобы $G(x, y) = 0$, если такого z не существует, определим функцию $\sigma \in \mathcal{E}^2$.

$$\left. \begin{array}{l} \sigma(x, 0) = x \\ \sigma(x, s(y)) = 0 \\ \sigma(x, y) \leq x \end{array} \right\} \begin{array}{l} \sigma(x, y) \text{ есть } x, \text{ если } y = 0, \\ \text{и } 0 \text{ в противном случае.} \end{array}$$

Заметим теперь, что $G(x, y) \div y$ есть 0 тогда и только тогда, когда $G(x, y) \leq y$. Следовательно, $\sigma(G(x, y), G(x, y) \div y)$ есть в точности нужная функция g . Отсюда g лежит в $\bar{F}_i^{n+1}$ по определению.

Следствие. Если $R \in (\bar{F}_i^{n+1})_*$ и $g \in \bar{F}_i$, то $h \in \bar{F}_i$, где

$$h(x_1, \dots, x_n, y_1, \dots, y_m) = \min z \leq g(y_1, \dots, y_m) [R(x_1, \dots, x_n, z)].$$

Доказательство. По определению $(\bar{F}_i^{n+1})_*$, R имеет характеристическую функцию $f \in \bar{F}_i^{n+1}$. Значит, функция j лежит в $\bar{F}_i^{n+1}$, где $j(x_1, \dots, x_n, y) = \min z \leq y [R(x_1, \dots, x_n, z)]$. Но h есть результат подстановки g вместо $(n+1)$ -го аргумента j , так что по определению 7 $h \in \bar{F}_i$.

Это следствие позволит нам заключить, что n -местная функция f лежит в $\bar{F}_i$, как только мы покажем, что

$$f(x) = U_{n,z}^i(x, \min y \leq b_{n,z}^i(x) [T_{n,z}^i(x, y)]),$$

где $U_{n,z}^i$, $b_{n,z}^i$ и $T_{n,z}^i$ лежат в $\bar{F}_i^{n+1}$, $\bar{F}_i$ и $(\bar{F}_i^{n+1})_*$ соответственно.

Определяем теперь $b_{n,z}^i$, $U_{n,z}^i$ и $T_{n,z}^i$ аналогично определениям $b_{n,z}$, $U_{n,z}$ и $T_{n,z}$, данным в конце предыдущего раздела, и показываем, что они лежат в нужных классах. Предположим, что мы рассматриваем вычисление некоторой функции из F_i на машине Тьюринга с номером z с k (внутренними) состояниями и тремя символами, которые могут появляться на ее ленте. Если даны аргументы $(x_1, \dots, x_n)$, то размер использованной ленты меньше, чем $f_{i-1}(K \max(x_1, \dots, x_n, 1))$, по лемме 3. Отсюда число мгновенных описаний вычисления меньше, чем

$$3^{f_{i-1}(K \max(x, 1))} \cdot k \cdot f_{i-1}(K \max(x, 1)).$$

Это в свою очередь меньше, чем величина

$$[f_i(K(x_1 + \dots + x_n + 1))]^2 \cdot k \cdot f_{i-1}(K(x_1 + \dots + x_n + 1)),$$

которую мы назовем $b_{n,z}^i$. Так как умножение есть функция из $\mathcal{E}^2$ и каждый $\bar{F}_i$ замкнут относительно подстановок в функции из $\mathcal{E}^2$, то каждый класс $\bar{F}_i$ должен быть замкнутым относительно умножения. Поэтому так как $f_1(K(x_1 + \dots + x_n + 1)) = [f_1(x_1) \dots f_1(x_n) \cdot 2]^K$, то $f_1(K(x_1 + \dots + x_n + 1))$ лежит в $\bar{F}_1$. Следовательно, $f_i(K(x_1 + \dots + x_n + 1))$ лежит в $\bar{F}_i$ по предложению в определении 7, гарантирующему, что $f_1 \circ f \in \bar{F}_i$ для любой $f \in \bar{F}_{i-1}$. Используя замкнутость $\bar{F}_i$ относительно умножения, получаем, что $b_{n,z}^i \in \bar{F}_i$.

Теперь мы можем определить функцию $H_{n,z}^i$, где $H_{n,z}^i(\mathbf{x}, y)$ есть номер мгновенного описания на y -м шаге машины с номером z , если задан вход $(x_1, \dots, x_n)$. Это определение есть в точности результат замещения границы $b_{n,z}$ в $H_{n,z}$ на $b_{n,z}^i$:

$$H_{n,z}^i(\mathbf{x}, 0) = \text{Init}_n(\mathbf{x}),$$

$$H_{n,z}^i(\mathbf{x}, s(y)) = \text{Yield}_z(H_{n,z}^i(\mathbf{x}, y)),$$

$$H_{n,z}^i(\mathbf{x}, y) \leq b_{n,z}^i(\mathbf{x}).$$

Так как $\text{Yield}_z \in \mathcal{E}^2$ и $b_{n,z}^i$, очевидно, лежит в $\bar{F}_i^{n+1}$, эта рекурсия определяет $H_{n,z}^i$ как функцию из $\bar{F}_i^{n+1}$ (по определению 7). Теперь определяем $U_{n,z}^i$ и $T_{n,z}^i$, где $U_{n,z}^i(\mathbf{x}, y)$ есть число на ленте в y -м мгновенном описании этого вычисления, и $T_{n,z}^i(\mathbf{x}, y)$ верно тогда, когда y есть количество мгновенных описаний в вычислении,

$$U_{n,z}^i(\mathbf{x}, y) = \text{Decode}(H_{n,z}^i(\mathbf{x}, y)),$$

$$T_{n,z}^i(\mathbf{x}, y) = \text{Fin}(H_{n,z}^i(\mathbf{x}, y)).$$

Так как функция $\text{Decode} \in \mathcal{E}^2$ и отношение $\text{Fin} \in \mathcal{E}_*^2$, имеем $U_{n,z}^i \in \bar{F}_i^{n+1}$ и $T_{n,z}^i \in (\bar{F}_i^{n+1})_*$. Таким образом,

$$\min y \leq b_{n,z}^i(\mathbf{x}) [T_{n,z}^i(\mathbf{x}, y)]$$

определяет функцию из $\bar{F}_i$ по следствию из теоремы 8. Получаем теорему 9.

Теорема 9. Функция f лежит в F_i тогда и только тогда, когда существует число z , такое, что

$$f(x) = U_{n,z}^i(x, \min y \leq b_{n,z}^i(x) [T_{n,z}^i(x, y)]).$$

Следствие. Для всех положительных целых¹⁾ i $F_i = \bar{F}_i$.

6. КЛАССЫ $(F_i)_*$ ОТНОШЕНИЙ

Мы знаем, что класс F_i строго включен в F_{i+1} и что $\bigcup_{i=1}^{\infty} F_i = \mathcal{E}$. Поэтому мы знаем также, что для соответствующих

классов отношений имеет место $(F_i)_* \subseteq (F_{i+1})_*$ и $\bigcup_{i=1}^{\infty} (F_i)_* = \mathcal{E}_*$.

Сейчас мы покажем, что в действительности включение $(F_i)_*$ в $(F_{i+1})_*$ является строгим. Таким образом, эти классы подразделяют класс элементарных множеств (и отношений) на классы согласно трудности разрешающей процедуры.

Часто говорят, например, что разрешающая процедура Тарского для теории первого порядка действительных чисел [9] сложнее, чем метод истинностных таблиц для исчисления высказываний. После того как мы покажем, что классы $(F_i)_*$ образуют подразбиение элементарных множеств, мы сможем уточнить это утверждение. Нужно заметить только, что метод истинностных таблиц принадлежит классу F_1 , а процедура Тарского (согласно расчетам автора) принадлежит²⁾ F_8 . Это кажется вполне достаточным подтверждением распространенного

¹⁾ Это следствие дает доказательство того, что $\mathcal{E} = \bigcup_{i=1}^{\infty} F_i$, без ссылок

на наши рассуждения в разд. 3. Мы показали в теореме 2, что $\mathcal{E} \subseteq \bigcup_{i=1}^{\infty} F_i$, и

в теореме 6, что $\bigcup_{i=1}^{\infty} \bar{F}_i \subseteq \mathcal{E}$, откуда и получаем нужное равенство.

²⁾ Это предполагает, конечно, подходящий перевод формул в числа, так как классы F_i состоят из числовых функций. Полезен такой перевод: предполагая, что язык имеет базисные символы $a_1, \dots, a_n$, интерпретируем символ a_i как целое i , а каждую формулу — как положительное целое число, n -ичной записью которого она является. Это устанавливает взаимно однозначное соответствие между конечными строками в алфавите $a_1, \dots, a_n$ и положительными целыми числами, так что эти множества можно отождествить.

сомнения относительно программирования последнего разрешающего метода и надежды на немедленные результаты¹⁾.

Прежде чем показать, что существует отношение, принадлежащее $(F_{i+1})_*$, но не принадлежащее $(F_i)_*$, заметим, что, хотя функция f_{i+1} не лежит в F_i , отношение $\lambda(x, y)[f_{i+1}(x) = y]$ лежит в $(F_i)_*$. Действительно, оно лежит в $\mathcal{E}^2$, что читатель может легко проверить. В этом смысле сложность f_i зависит почти исключительно от скорости ее роста. Если дано достаточно ленты, чтобы написать ответ, то вычисление лежит в $\mathcal{E}^2$. Однако характеристическая функция, принадлежащая F_{i+1} и не принадлежащая F_i , является в строгом смысле более сложной, чем функция из F_i . Дополнительная длина ленты в действительности используется для вычислительных целей, а не просто для того, чтобы записать большой ответ.

Чтобы показать, что $(F_i)_* \neq (F_{i+1})_*$, мы строим в F_{i+1} характеристическую функцию Φ_i со следующим свойством. $\Phi_i(z, x)$ принимает значение 0, если $f(x) = 0$, где $f(x)$ — функция, вычисляемая на машине Тьюринга с номером z , и $\Phi_i(z, x)$ равна 1 в противном случае. Далее, с помощью обычной процедуры диагонализации показываем, что $\Phi_i \notin F_i$, так что отношение, характеристическая функция которого есть Φ_i , лежит в $(F_{i+1})_*$, но не лежит в $(F_i)_*$.

Чтобы определить Φ_i , мы используем арифметизацию из приложения 3 для получения нового T -предиката и функции U , которые являются функциями от z . Начнем с определения ограничивающей функции b_n^i из F_{i+1} для числа шагов в вычислении некоторой n -местной функции из F_i . Мы могли бы сделать это, положив просто

$$b_n^i(z, \mathbf{x}) = b_{n,z}^i(\mathbf{x}) = [f_i(K \cdot (x_1 + \dots + x_n + 1))]^2 \cdot k \cdot f_{i-1}(K \cdot (x_1 + \dots + x_n + 1)),$$

где K и k суть числа, зависящие только от z . Так как каждой машине Тьюринга соответствует бесконечно много гёделевых номеров²⁾, мы можем выбрать гёделев номер z так, что K и k будут оба меньше z . Следовательно, мы можем заменить каждое из них на z и использовать для ограничивающей функции определение

$$b_n^i(z, \mathbf{x}) = [f_i(z \cdot (x_1 + \dots + x_n + 1))]^2 \cdot z \cdot f_{i-1}(z \cdot (x_1 + \dots + x_n + 1)).$$

¹⁾ Отсюда не следует, конечно, что для этого множества нет разрешающих процедур в F_1 , а следует только, что процедура, предложенная в [9], не лежит в F_1 , т. е. множество верных формул принадлежит F_{8*} , но не исключена возможность, что оно лежит в F_{i*} для некоторого $i < 8$.

²⁾ Наша арифметизация допускает, чтобы список состояний машины Тьюринга содержал повторения.

Так как F_1 замкнут относительно умножения, $z(x_1 + \dots + x_n + 1)$ лежит в F_1 . Таким образом, $f_i(z \cdot (x_1 + \dots + x_n + 1))$ принадлежит F_{i+1} по лемме 2, и, значит, $b_n^i \in F_{i+1}$.

Определим теперь H_n^i , T_n^i , U_n^i и функцию Ψ_n^i следующим образом:

$$\begin{aligned} H_n^i(z, \mathbf{x}, 0) &= \text{Init}_n(\mathbf{x}), \\ H_n^i(z, \mathbf{x}, s(y)) &= \text{Yield}(H_n^i(z, \mathbf{x}, y), z), \\ H_n^i(z, \mathbf{x}, y) &\leq b_n^i(z, \mathbf{x}), \\ U_n^i(z, \mathbf{x}, y) &= \text{Decode}(H_n^i(z, \mathbf{x}, y)), \\ T_n^i(z, \mathbf{x}, y) &= \text{Fin}(H_n^i(z, \mathbf{x}, y)), \\ \Psi_n^i(z, \mathbf{x}) &= U_n^i(z, \mathbf{x}, \min y \leq b_n^i(z, \mathbf{x}) [T_n^i(z, \mathbf{x}, y)]). \end{aligned}$$

Если $f \in F_i$, то по лемме 3 существует число K , такое, что $a_f(\mathbf{x}) < f_{i-1}(K \cdot \max(\mathbf{x}, 1))$. Тогда по сказанному выше f вычисляется на машине Тьюринга, гёделев номер z которой больше K . Число шагов в вычислении $f(\mathbf{x})$ меньше, чем $b_{n,z}^i(\mathbf{x})$, так что для любой функции $f \in F_i$ существует число z , такое, что

$$f(\mathbf{x}) = \Psi_n^i(z, \mathbf{x}).$$

Определим теперь характеристическую функцию Φ_i (от двух аргументов), полагая

$$\Phi_i(z, x) = \sigma(1, 1 \div \Psi_1^i(z, x)),$$

так что имеем $\Phi_i(z, x) = \min(1, f(x))$, если машина с номером z вычисляет одноместную функцию f . Далее, $\Phi_i \in F_{i+1}$, так как Ψ_n^i была построена в F_{i+1} .

Чтобы показать, что $\Phi_i \notin F_i$, предположим противное. Тогда функция Θ также лежит в F_i , где $\Theta(x) = 1 \div \Phi_i(x, x)$, и, так как Θ есть одноместная функция, принимающая значения 0 или 1, существует z_0 , для которого $\Theta(x) = \Psi_1^i(z_0, x) = \Phi_i(z_0, x)$.

В частности, $\Theta(z_0) = \Phi_i(z_0, z_0)$, но по определению $\Theta(z_0) = 1 \div \Phi_i(z_0, z_0) \neq \Phi_i(z_0, z_0)$, что является противоречием. Это доказывает сформулированную ниже теорему 10, так как отношение, характеристической функцией которого является Φ_i , лежит в $(F_{i+1})_*$, но не лежит в $(F_i)_*$.

Теорема 10. Для каждого положительного целого i $(F_i)_*$ строго включен в $(F_{i+1})_*$.

ПРИЛОЖЕНИЕ 1

ЭЛЕМЕНТАРНЫЕ ОПРЕДЕЛЕНИЯ MR, CORN И FIN

В [1] вводится соглашение обозначать S_0 символом β , а 1 вводится для S_1 . Условимся обозначать 0 через S_2 . Так как символу S_i соответствует в [1] гёделев номер 2^{4i+7} , то β , 1 и 0 получают гёделевы номера 2^7 , 2^{11} и 2^{15} соответственно. Следуя гёделевой нумерации, использованной в [1], имеем, что каждой ленте, состоящей из β , 1 и 0, соответствует единственный номер; например, лента $10\beta 1$ имеет гёделев номер $2^{11} \cdot 3^{15} \cdot 5^7 \cdot 7^{11}$. Используя функцию сочленения $*$, введенную в [1], мы можем записать этот номер как $2^{11} * 2^{15} * 2^7 * 2^{11}$. (Мы будем свободно использовать здесь эти соглашения.)

Чтобы определить MR, введем сначала элементарные множества (1-местные отношения) Zeros и Ones и элементарную функцию S. Первые два говорят сами за себя, последняя, будучи применена к гёделеву номеру ленты, представляющей n , будет выдавать гёделев номер ленты, представляющей $n+1$:

$$\text{Zeros}(x) \leftrightarrow (\forall y)_{\leq x} [\mathcal{L}(y) = 1 \wedge (\exists z, w)_{\leq x} [z * y * w = x] \rightarrow y = 2^{15}],$$

$$\text{Ones}(x) \leftrightarrow (\forall y)_{\leq x} [\mathcal{L}(y) = 1 \wedge (\exists z, w)_{\leq x} [z * y * w = x] \rightarrow y = 2^{11}],$$

$$\begin{aligned} S(x) = \min y \leq (x * x * x) \{ & (\exists z)_{\leq x} [z * 2^{15} = x \wedge z * 2^{11} = y] \vee \\ & \vee (\exists z, w, v)_{\leq y} [z * 2^{15} * w = x \wedge \text{Ones}(w) \wedge \text{Zeros}(v) \wedge \\ & \wedge \mathcal{L}(w) = \mathcal{L}(v) \wedge z * 2^{11} * v = y] \vee \\ & \vee (\exists v)_{\leq y} [\text{Ones}(x) \wedge \text{Zeros}(v) \wedge \mathcal{L}(x) = \mathcal{L}(v) \wedge 2^{11} * v = y] \}. \end{aligned}$$

Здесь $\mathcal{L}(x)$ есть число символов в последовательности с гёделевым номером x (как определяется на стр. 58 в [1]).

Определим теперь MR при помощи ограниченной рекурсии:

$$\text{MR}(0) = 2^{11},$$

$$\text{MR}(n+1) = S(\text{MR}(n)),$$

$$\text{MR}(n) \leq (\text{Pr}(n+1)^{15})^{n+1}.$$

Тогда Corn определяется просто как MR^{-1} , т. е.

$$\text{Corn}(x) = \min y \leq x [\text{MR}(y) = x].$$

Сделав одно изменение в определении Fin, получаем

$$\begin{aligned} \text{Fin}(x, z) \leftrightarrow & \text{ID}(x) \wedge \text{TM}(z) \wedge (\exists F, G, r, s)_{\leq x} \{ (x = F * 2^r * 2^s * G) \wedge \\ & \wedge \text{IC}(r) \wedge \text{Al}(s) \wedge (\exists y)_{\leq x} [\text{MR}(y) = F * 2^s * G] \wedge \\ & \wedge (\forall n)_{\leq \mathcal{L}(z)} [(1Gl(nGlz) \neq r) \vee (2Gl(nGlz) \neq s)] \}. \end{aligned}$$

ПРИЛОЖЕНИЕ 2

МАШИНЫ ТЬЮРИНГА И КОНЕЧНЫЕ АВТОМАТЫ

В этом приложении мы даем формальные определения машины Тьюринга и вычисления на такой машине. Затем мы определяем «вычисление на машине Тьюринга, представляющей собой конечный автомат», и класс F числовых функций, вычисляемых таким образом. Мы завершаем это приложение доказательством некоторых свойств F , которые необходимы для основной части этой статьи.

Определение 1. Алфавит — это конечное множество, элементы которого называются символами и которое всегда содержит в качестве элемента символ β (так называемый пустой символ).

Определение 2. Машина Тьюринга Z в алфавите A есть четверка (S, m, s_0, f) , где S — конечное множество, s_0, f — элементы S и $m: A \times (S - \{f\}) \rightarrow A \times S \times \{1, -1, 0\}$. Мы называем S множеством состояний Z , s_0 — начальным состоянием, f — заключительным состоянием и m — функцией перехода.

Определение 3. Если дана машина Тьюринга $Z = (S, m, s_0, f)$ в алфавите A , определяем *мгновенное описание* Z как тройку (t, s, p) , где t — конечная последовательность элементов из A , p — положительное целое, не превосходящее длины t , и s — элемент из S . Мы называем t лентой машины Z , p — номером воспринимаемой ячейки и s — состоянием Z .

Лента машины Z может удлиняться вправо и имеет фиксированную самую левую ячейку, обозначенную числом 1. Возможность удлинения ленты вправо является причиной появления трех последних пунктов в определении операции $\rightarrow$, приведенном ниже. Эти пункты разрешают прибавление дополнительной ячейки с пустым символом в ней, если машина сходит с правого конца ленты, а также обеспечивают выбрасывание из рассмотрения самого правого пустого символа, если машина его больше не воспринимает.

Определение 4. Если дана машина Тьюринга $Z = (S, m, s_0, f)$ в алфавите A , определяем операцию $\rightarrow$ между мгновенными описаниями Z следующим образом. $X \rightarrow Y(Z)$ означает, что выполнено одно из следующих условий (здесь X и Y — мгновенные описания Z и a_i, b_i — символы алфавита A для всех положительных целых i):

(1) $X = (a_1 \dots a_n, s, p)$ и $Y = (b_1 \dots b_n, s', p')$, причем $a_j = b_j$ для всех $j \neq p$, $m(a_p, s) = (b_p, s', p' - p)$ и либо $p < n$, либо $p = p' = n$.

(2) $X = (a_1 \dots a_{n-1}a_n, s, n)$ и $Y = (a_1 \dots a_{n-1}b_n\beta, s', n+1)$, где $m(a_n, s) = (b_n, s', 1)$.

(3) $X = (a_1 \dots a_{n-1}a_n, s, n)$ и $Y = (a_1 \dots a_{n-1}b_n, s', n-1)$, где $m(a_n, s) = (b_n, s', -1)$ и $b_n \neq \beta$.

(4) $X = (a_1 \dots a_{n-1}a_n, s, n)$ и $Y = (a_1 \dots a_{n-1}, s', n-1)$, где $m(a_n, s) = (\beta, s', -1)$.

Определение 5. *Вычисление* на машине Тьюринга Z в алфавите A есть конечная последовательность $X_1, \dots, X_q$ мгновенных описаний Z , такая, что для всех $i = 1, \dots, q-1$ $X_i \rightarrow X_{i+1}(Z)$ и $X_q = (t, f, p)$ для некоторой конечной последовательности t элементов A и некоторого целого p . Мы говорим, что X_1 *начинает* это вычисление и что X_q — *результат* X_1 .

Определение 6. Если дано подмножество D множества T_A всех конечных последовательностей элементов алфавита A , то говорят, что функция $\varphi: D \rightarrow T_A$ *вычисляется* на машине Тьюринга Z в алфавите A , если выполняются следующие условия. Для каждой последовательности $t \in D$ существует вычисление на Z , начинающееся с $(t, s_0, 1)$, причем результатом $(t, s_0, 1)$ является $(\varphi(t), f, p)$ для некоторого p .

Кодируя числовые функции в двоичной системе счисления, зафиксируем алфавит $\{0, 1, \beta\}$ как алфавит, в котором работают все машины Тьюринга, вычисляющие числовые функции. Мы будем часто опускать слова «в алфавите $\{0, 1, \beta\}$ », когда будет идти речь о машинах Тьюринга, вычисляющих числовые функции. Определим теперь вычисление числовых функций следующим образом.

Определение 7. Будем говорить, что машина Тьюринга Z в алфавите $\{0, 1, \beta\}$ *вычисляет* функцию f от n неотрицательных целых аргументов с неотрицательными целыми значениями, если она вычисляет функцию $f': D \rightarrow T_{\{0, 1\}}$, где

(I) D есть множество всех строк $\bar{a}_1\beta\bar{a}_2\beta \dots \beta\bar{a}_n$ для n -ок $(a_1, a_2, \dots, a_n)$ из области определения f ,

(II) $f'(\bar{a}_1\beta \dots \beta\bar{a}_n)$ определяется как $\overline{f(a_1, \dots, a_n)}$.

Здесь $\bar{x}$ означает двоичную запись неотрицательного целого x .

Это завершает наши формальные определения машин Тьюринга и вычислений на таких машинах. В приложении 3 мы приводим арифметизацию этих машин и получаем некоторые результаты, нужные для разд. 4 — 6 этой статьи.

Обратимся теперь к конечным автоматам. Здесь нам удобнее всего рассматривать их как машины Тьюринга, которые вычисляют несколько отличным (и очень ограниченным) способом. Вычисление на такой машине всегда начинается с крайней правой ячейки входной ленты, и далее машина сдвигается на

одну ячейку влево на каждом шаге вычисления. В таком вычислении машина может удлинять ленту *влево* на столько, на сколько это необходимо. Поэтому здесь удобно считать, что ячейки занумерованы справа налево. Мы уточняем эти замечания в следующих определениях.

Определение 8. Если дана машина Тьюринга $Z = (S, m, s_0, f)$ в алфавите A , мы определяем операцию $\rightarrow_F$ между мгновенными описаниями Z следующим образом. $X \rightarrow_F Y(Z)$ означает, что выполняется одно из следующих условий, где X и Y — мгновенные описания Z и a_i и b_i — элементы A .

(1) $X = (a_n a_{n-1} \dots a_1 a_0, s, p)$ и $Y = (b_n b_{n-1} \dots b_1 b_0, s', p+1)$, где $p < n$, $b_j = a_j$ для всех $j \neq p$ и $m(a_p, s) = (b_p, s', -1)$.

(2) $X = (a_n a_{n-1} \dots a_1 a_0, s, n)$ и $Y = (b_n a_{n-1} \dots a_1 a_0, f, n+1)$, где $m(a_n, s) = (b_n, f, -1)$.

(3) $X = (a_n a_{n-1} \dots a_1 a_0, s, n)$ и $Y = (\beta b_n a_{n-1} \dots a_1 a_0, s', n+1)$, где $m(a_n, s) = (b_n, s', -1)$ и $s' \neq f$.

Определение 9. F -вычисление на машине Тьюринга в алфавите A есть конечная последовательность $X_1, \dots, X_q$ мгновенных описаний Z , такая, что

(I) $X_i \rightarrow_F X_{i+1}(Z)$ для всех $i = 1, \dots, q-1$ и

(II) $X_1 = (t, s_0, 0)$ и $X_q = (t', f, l(t'))$,

где t и t' — конечные последовательности элементов A и $l(t')$ означает длину последовательности t' .

Мы говорим, что X_1 *начинает* вычисление, а X_q является *результатом* X_1 .

Определение 10. Если дано подмножество D множества T_A всех конечных последовательностей элементов алфавита A , говорят, что функция $\Phi: D \rightarrow T_A$ *вычисляется на машине Тьюринга Z , представляющей собой конечный автомат* в алфавите A , если выполняются следующие условия. Для каждого $t \in D$ существует F -вычисление на Z , начинающееся с $(t, s_0, 0)$, причем $(\Phi(t), f, l(\Phi(t)))$ есть результат $(t, s_0, 0)$.

Когда функция Φ вычисляется на машине Тьюринга Z , представляющей собой конечный автомат, мы говорим о Z как о конечном автомате и о Φ как о функции, вычисляемой на этом конечном автомате. Таким образом, когда мы ссылаемся на конечные автоматы, мы подразумеваем просто машины Тьюринга, которые выполняют только F -вычисления; когда мы говорим о вычислении на конечном автомате Z , мы имеем в виду F -вычисление на машине Тьюринга Z . Мы будем свободно использовать эту сокращенную терминологию на протяжении всей статьи.

Теорема 1. Если Φ вычисляется конечным автоматом Z с k *незаключительными состояниями*, тогда для каждого аргумента

t из области определения Φ длина $\Phi(t)$ не больше, чем $k+l(t)$, где $l(t)$ — длина ленты t .

Доказательство. По определению 8, после того как Z прочтет все символы t , он продолжает двигаться влево, читая пустые символы на каждом шаге (пока не достигнет заключительного состояния и не остановится). Однако если он попадает дважды в одно и то же состояние, в то время как читает эти пустые символы, то он должен заикнуться и будет работать бесконечно долго. Так как k есть число различных незаключительных состояний Z и t принадлежит области определения Φ , то Z должен попасть в заключительное состояние f не позже, чем через k шагов после прочтения последнего символа t . Таким образом, длина $\Phi(t)$ не превосходит $k+l(t)$.

Поэтому, а также из-за других серьезных ограничений, наложенных на функции, вычислимые конечными автоматами, принятые методы кодирования числовых входов отличаются от кодирования, указанного в нашем определении 7 для машин Тьюринга. (См., например, [2, стр. 31].)

Чтобы ввести n числовых аргументов в машину Тьюринга, каждое число записывают в двоичной системе, и эти n двоичных строк пишут по порядку на ленте, разделяя их пустыми символами. Можно представить себе, что числа в машину подаются последовательно, одно за другим. Напротив, в конечный автомат числа подаются параллельно; последняя значащая цифра каждого из n чисел читается на первом шаге вычисления, следующая значащая цифра каждого числа — на втором шаге и т. д. Чтобы упростить обозначения, ограничим наши рассуждения функциями одного или двух аргументов. (Единственной дополнительной трудностью, возникающей при рассмотрении общего случая, является неудобство обозначений.)

Предполагается, что конечные автоматы, вычисляющие функции одного аргумента, работают в алфавите $A_1 = \{\beta, 0, 1\}$. Называем строку символов из этого алфавита **кодом** неотрицательного целого n , если она имеет вид $\beta \dots \beta \bar{n}$, где $\bar{n}$ — (единственная) запись n в двоичной системе счисления. Так, например, $\beta\beta 101$ и 101 суть коды числа пять, но ни 0101 , ни 101β не являются кодами. Определяем теперь вычисление числовых функций одного переменного.

Определение 11. Говорят, что конечный автомат в алфавите A_1 **вычисляет функцию f от неотрицательных целых с целыми неотрицательными значениями**, если он вычисляет некоторую функцию $f': D \rightarrow T_{\{0,1\}}$, такую, что

(I) D есть множество всех $\bar{n}$ записей (в двоичной системе счисления) неотрицательных целых n ;

(II) $f'(\bar{n})$ есть код $f(n)$.

Прежде чем определять вычисление числовой функции двух переменных, мы введем еще два соглашения. Во-первых, будем предполагать, что конечный автомат, вычисляющий такую функцию, работает в алфавите

$$A_2 = \left\{ \begin{matrix} \beta, & \beta, & \beta, & 0, & 1, & 0, & 0, & 1, & 1 \\ \beta, & 0, & 1, & \beta, & \beta, & 0, & 1, & 0, & 1 \end{matrix} \right\}.$$

Мы будем иногда писать

β вместо пустого символа $\begin{smallmatrix} \beta \\ \beta \end{smallmatrix}$,

0 вместо символа $\begin{smallmatrix} \beta \\ 0 \end{smallmatrix}$,

1 вместо $\begin{smallmatrix} \beta \\ 1 \end{smallmatrix}$.

Таким образом, мы можем говорить об A_1 как о подмножестве A_2 . Во-вторых, для каждой пары (m, n) неотрицательных целых определяем строки m_n и n_m из элементов алфавита A_1 следующим образом. Предположим (без ограничения общности), что запись числа n в двоичной системе содержит на k знаков больше, чем запись m ($k \geq 0$). Тогда n_m есть просто $\bar{n}$, двоичная запись n , а m_n есть $\beta \dots \beta \bar{m}$ с k символами β . Таким образом, m_n и n_m суть коды m и n соответственно; они имеют одинаковую длину, и один из них совпадает с двоичной записью числа. Пусть теперь дана пара (m, n) , пусть $m_n = a_1 a_2 \dots a_p$ и $n_m = b_1 b_2 \dots b_p$, где a_i и b_i — элементы A_1 . Тогда мы обозначаем через $(\overline{m, n})$ ленту

$$\left\{ \begin{matrix} a_1 a_2 \dots a_p \\ b_1 b_2 \dots b_p \end{matrix} \right\}$$

в алфавите A_2 .

Определение 12. Будем говорить, что конечный автомат Z в алфавите A_2 вычисляет функцию f от пар неотрицательных целых с неотрицательными целыми значениями, если он вычисляет некоторую функцию $f': D \rightarrow T_{A_1}$, такую, что

(I) D есть множество всех лент $(\overline{m, n})$ для всех пар (m, n) неотрицательных целых;

(II) $f'(\overline{m, n})$ есть код $f(m, n)$. (В этом определении мы отождествляем алфавит A_1 с подмножеством A_2 .)

Подобным же образом можно определить вычисления на конечных автоматах числовых функций любого числа переменных. Мы предполагаем, что эти определения проведены, и обозна-

чаем через F класс всех числовых функций любого числа переменных, которые вычисляются на конечных автоматах.

В качестве примера рассмотрим функцию следования и сложение. Функция следования вычисляется на конечном автомате Z , который действует следующим образом. При входе 0 Z выдает $\beta 1$, при входе 1 Z выдает 10, при входе 10 Z выдает $\beta 11$ и т. д. (Пустые символы в выходе появляются потому, что вычисление не может завершиться, пока не будет прочитан пустой символ слева от входа.) Построение такой машины предоставляется читателю. Сложение вычисляется на автомате, у которого множество пар «вход — выход» содержит, например, такие пары:

$$\left\{ \begin{matrix} 1 \\ 0 \end{matrix}, \beta 1 \right\}, \quad \left\{ \begin{matrix} 1 \\ 1 \end{matrix}, 10 \right\}, \quad \left\{ \begin{matrix} \beta 1 \\ 10 \end{matrix}, \beta 11 \right\}, \quad \left\{ \begin{matrix} \beta 1 \\ 11 \end{matrix}, 100 \right\}$$

и

$$\left\{ \begin{matrix} \beta \beta 101 \\ 10011 \end{matrix}, \beta 11000 \right\}.$$

Можно легко построить также конечный автомат, вычисляющий произвольную константную функцию C_q^n от n аргументов или функцию выбора U_i^n от n аргументов, определенные, например, в [4, стр. 197].

Теперь мы хотим установить некоторые свойства класса F числовых функций, вычислимых на конечных автоматах. Желаемыми свойствами являются свойство замкнутости относительно композиции и явных преобразований и свойство линейной ограниченности в подходящем смысле. Приведем непосредственное доказательство замкнутости F относительно композиции. Мы построим конечный автомат, вычисляющий $g \circ h$, где $(g \circ h)(y) = g(h(y))$, если даны автоматы, вычисляющие g и h . Мы опустим детали доказательства того, что этот конечный автомат действительно работает, так же как и обобщение на случай композиции функций произвольного числа переменных.

Предположим, что одноместные числовые функции g и h вычисляются на конечных автоматах $Z_g = (S_g, m_g, s_{g0}, f_g)$ и $Z_h = (S_h, m_h, s_{h0}, f_h)$ соответственно. Чтобы сократить определение автомата для композиции $g \circ h$, мы вводим следующие сокращения. Для $a \in A_1$, $s_h \in S_h$ и $s_g \in S_g$:

(I) пусть $a_h = a$ и $s'_h = f_h$, если $s_h = f_h$; в противном случае пусть $a_h = x$ и $s'_h = y$, где $m_h(a, s_h) = (x, y, -1)$;

(II) пусть $a' = a_h$ и $s'_g = f_g$, если $s_g = f_g$; в противном случае пусть $a' = u$ и $s'_g = v$, где $m_g(a_h, s_g) = (u, v, -1)$.

Теперь определяем функцию

$$m': A_1 \times [(S_g \times S_h) - \{(f_g, f_h)\}] \rightarrow A_1 \times (S_g \times S_h) \times \{-1\},$$

полагая $m'(a, (s_g, s_h)) = (a', (s'_g, s'_h), -1)$, и, наконец, полагаем

$$Z_{g \circ h} = (S_g \times S_h, m', (s_{g0}, s_{h0}), (f_g, f_h)).$$

Чтобы убедиться, что $Z_{g \circ h}$ вычисляет $g \circ h$, предположим, что $Z_{g \circ h}$ находится в состоянии (s_g, s_h) на n -м шаге вычисления, начинающегося с ленты t , и что Z_h , будучи применен к t , выдает ленту t' . Тогда s_h есть состояние, в которое попадает Z_h на n -м шаге вычисления, начатого с ленты t , а s_g есть состояние, в которое попадает Z_g на n -м шаге вычисления, начатого с ленты t' . Индуктивное доказательство того, что $Z_{g \circ h}$ вычисляет $g \circ h$, базируется непосредственно на этом замечании и предоставляется читателю.

Замкнутость F относительно явных преобразований является следствием предыдущих результатов. Пусть $f(x_1, \dots, x_n) = g(\xi_1, \dots, \xi_k)$. Если $\xi_i = x_j$, определяем функцию f_i , полагая $f_i(x_1, \dots, x_n) = U_j^n(x_1, \dots, x_n)$, и, если ξ_i есть константа, полагаем $f_i(x_1, \dots, x_n) = C_{\xi_i}^n(x_1, \dots, x_n)$. Каждая f_i лежит в F по предыдущим рассуждениям. Так как

$$f(x_1, \dots, x_n) = g(f_1(x_1, \dots, x_n), \dots, f_k(x_1, \dots, x_n))$$

и F замкнут относительно композиции, то $f \in F$.

Так как множество линейных функций можно определить как замыкание сложения, функций выбора и константных функций относительно композиции и явных преобразований, то мы знаем теперь, что каждая линейная функция лежит в F . Хотя не каждая функция из F является линейной, любая функция из F ограничена некоторой линейной функцией в смысле, указанном следующей теоремой.

Теорема 2. Если $f(x_1, \dots, x_n)$ из F , то существует такое неотрицательное целое K , что для всех $x_1, \dots, x_n$ $f(x_1, \dots, x_n) < K \cdot \max(x_1, \dots, x_n, 1)$.

Доказательство. Для всякого целого y обозначим через $l(y)$ длину двоичной записи y . Сразу получаем, что $y < 2^{l(y)} \leq 2 \cdot \max(y, 1)$. С помощью определений 11 и 12 (и их обобщений) устанавливаем, что для конечного автомата запись n -ки $(x_1, \dots, x_n)$ есть лента длины $l[\max(x_1, \dots, x_n, 1)]$. Введем сокращение x для $x_1, \dots, x_n$. Теперь по теореме 1

этого приложения существует такое k , что $l[f(x)] < l[\max(x, 1)] + k$. Теорема следует из этого неравенства, ибо $f(x) < 2^{l[f(x)]} < 2^k \cdot 2^{l[\max(x, 1)]} \leq 2^{k+1} \cdot \max(x, 1)$.

ПРИЛОЖЕНИЕ 3

АРИФМЕТИЗАЦИЯ

В этом приложении мы хотим провести арифметизацию машин Тьюринга, вычисляющих числовые функции, и сделать это в пределах весьма ограниченного класса функций. Мы должны отказаться от представления последовательностей степенями простых чисел и, как это было указано в основной части статьи, обратиться к более экономному способу. Мы уже решили действовать с сочленением строк символов, ибо кажется, что этот процесс подсказан самими операциями машин Тьюринга. Так, продолжая, мы будем следовать Смультяну [8] и используем двужначную¹⁾ запись целых положительных чисел строками из единиц и двоек. Эта запись имеет техническое преимущество перед двоичной, так как устанавливает взаимно однозначное соответствие между строками конечной длины и (натуральными) числами. Легко видеть, что всякое целое положительное число имеет единственное представление в виде суммы членов вида $a_i \cdot 2^i$, где a_i есть 1 или 2, и числу $a_n \cdot 2^n + \dots + a_{n-1} \cdot 2^{n-1} + \dots + a_0 \cdot 2^0$ соответствует строка $a_n a_{n-1} \dots a_0$.

В [8] Смультян исследовал класс отношений, которые он называл рудиментарными отношениями, над целыми положительными числами. Это наименьший класс отношений, содержащий (двужначное) сочленение и замкнутый относительно явных преобразований, булевых операций и ограниченной квантификации. Здесь сочленение есть отношение, определенное следующим образом: z называется сочленением x и y (пишем $z = xy^2$), если двужначная запись z есть $x_n x_{n-1} \dots x_0 y_m y_{m-1} \dots y_0$, где $x_n \dots x_0$ и $y_m \dots y_0$ суть двужначные записи x и y . Мы хотим применить в нашей арифметизации рудиментарные отношения и, чтобы это осуществить, покажем сейчас, что всякое рудиментарное отношение содержится в $\mathcal{E}_*^2$.

Гжегорчик [3] показал, что класс $\mathcal{E}_*^2$ замкнут относительно явных преобразований, булевых операций и ограниченной квантификации, а Смультян [8, стр. 30—31, 77—80] доказал, что

¹⁾ В оригинале «dyadic». — Прим. перев.

²⁾ Везде в этом приложении, если нам необходимо сослаться на произведение x и y , мы будем обозначать его через $x \cdot y$; мы резервируем обозначение xy для сочленения.

всякий класс, замкнутый относительно этих операций и содержащий отношения сложения и умножения, будет содержать и сочленение. В [3] показано также, что $\mathcal{E}^2$ содержит функции сложения и умножения. Чтобы установить наш результат, мы доказываем следующую теорему.

Теорема 1. Если функция f лежит в $\mathcal{E}^2$, то отношение F принадлежит $\mathcal{E}_*^2$, где $F(x_1, \dots, x_n) \equiv f(x_1, \dots, x_{n-1}) = x_n$.

Доказательство. Предположим, что $f \in \mathcal{E}^2$, определяем отношения R и S , полагая $R(x, y) \equiv f(x) \dot{-} y = 0$ и $S(x, y) \equiv y \dot{-} f(x) = 0$. R и S лежат в $\mathcal{E}_*^2$, так как их характеристические функции g и h можно определить так: $g(x, y) = 1 \dot{-} (1 \dot{-} (f(x) \dot{-} y))$ и $h(x, y) = 1 \dot{-} (1 \dot{-} (y \dot{-} f(x)))$. Но тогда $R \cap S$ есть отношение из $\mathcal{E}_*^2$, и оно в точности совпадает с нужным отношением, так как

$$(R \cap S)(x, y) \equiv R(x, y) \wedge S(x, y) \equiv f(x) \leq y \wedge y \leq f(x) \equiv f(x) = y.$$

Следствие¹⁾. Класс рудиментарных отношений включен в класс $\mathcal{E}_*^3$.

Теперь мы можем провести нашу арифметизацию, используя только рудиментарные отношения, т. е. определяя отношения из сочленения при помощи явных преобразований, булевых операций и ограниченной квантификации. Определяем сначала некоторые отношения, которые будут использоваться в дальнейшем:

$$\begin{aligned} x_1 x_2 \dots x_n = y &\equiv (\exists z_2, z_3, \dots, z_{n-1})_{\leq y} \\ &[z_2 = x_1 x_2 \wedge z_3 = z_2 x_3 \wedge \dots \wedge z_{n-1} = z_{n-2} x_{n-1} \wedge y = z_{n-1} x_n], \\ xBy &\equiv x = y \vee (\exists z)_{\leq y} [xz = y] \quad (\text{читается „} x - \text{начало } y \text{“}), \\ xEy &\equiv x = y \vee (\exists z)_{\leq y} [zx = y] \quad (\text{читается „} x - \text{конец } y \text{“}), \\ xPy &\equiv xBy \vee xEy \vee (\exists z_1, z_2)_{\leq y} [z_1 x z_2 = y] \quad (\text{читается „} x - \text{часть } y \text{“}), \end{aligned}$$

$\text{tally}(x) \equiv \sim 2Px$ (читается „ x есть подсчет“, т. е. строка из 1),

$$l(x) = l(y) \equiv (\exists z)_{\leq x} [\text{tally}(z) \wedge z \leq x \leq 2z \wedge z \leq y \leq 2z] \\ (\text{читается „длина } x \text{ равна длине } y \text{“}),$$

$$l(x) = 5 \cdot l(y) \equiv (\exists z)_{\leq x} [\text{tally}(z) \wedge (\exists w)_{\leq y} [\text{tally}(w) \wedge l(y) = \\ = l(w) \wedge l(z) = l(x) \wedge wwwwww = z]].$$

¹⁾ Легко видеть, что отношение допускается линейно ограниченным автоматом в смысле [5] тогда и только тогда, когда его характеристическая функция лежит в F' . Таким образом, это следствие вместе с теоремой 4 дает результат, изложенный в [5], о том, что каждое рудиментарное отношение допускается некоторым линейно ограниченным автоматом.

После этих предварительных построений мы можем приступить к арифметизации машин Тьюринга (определенных в приложении 2), т. е. мы можем закодировать описания машин Тьюринга и вычисления на машинах Тьюринга числами (записанными в двузначном обозначении) и числовыми функциями соответственно. Тогда мы сможем выражать свойства машин Тьюринга через свойства функций, как это требуется в разд. 4—6. Для этого мы сопоставим каждому состоянию машины Тьюринга $Z = (S, m, s_0, f)$ число вида $11 \dots 11$. В частности, мы условимся сопоставлять число 1 начальному состоянию s_0 и число 11 заключительному состоянию f .

Мы обозначаем три символа алфавита 0, 1, β , которые могут встретиться на ленте, числами 1, 2 и 11 соответственно и обозначаем элементы 1, -1 и 0, принадлежащие области значений функции m , при помощи 1, 2 и 11 соответственно (т. е. 1 обозначает сдвиг ленты влево, 2 — сдвиг вправо и 11 — отсутствие сдвига). Для удобства запоминания различных чисел, обозначающих состояния, символы на ленте и команды сдвига, введем следующие сокращения:

$St(x) \equiv \text{tally}(x)$	(x есть номер состояния),
$s_0(x) \equiv x = 1$	(x есть номер состояния s_0),
$f(x) \equiv x = 11$	(x есть номер состояния f),
$A(x) \equiv x = 1 \vee x = 2 \vee x = 11$	(x есть номер символа алфавита),
$L(x) \equiv x = 1$	(x есть число, обозначающее сдвиг влево),
$R(x) \equiv x = 2$	(x обозначает сдвиг вправо),
$N(x) \equiv x = 11$	(x обозначает отсутствие сдвига ленты).

Полное описание машины Тьюринга $Z = (S, m, s_0, f)$ (в алфавите $\{0, 1, \beta\}$) получается перечислением s_0, f и m , ибо, зная m , мы знаем и S , так как $m: A \times (S - \{f\}) \rightarrow A \times S \times \{1, -1, 0\}$. Поскольку у нас есть соглашение относительно обозначений для s_0 и f , нам необходимо лишь указать m , чтобы полностью определить Z . Мы задаем m таблицей пятерок (a, s, a', s', p) , такой, что $m(a, s) = (a', s', p)$. Таким образом, чтобы определить номер машины Тьюринга Z , мы определяем вначале множество Quint номеров пятерок (a, s, a', s', p) ; множество Table чисел, являющихся номерами таблиц (списков) пятерок (a, s, a', s', p) ; отношение Quint, которое верно для x и y , если x является номером пятерки, встречающейся в таблице с номером y ; множество Fn номеров таблиц, являющихся функциями (т. е. таблиц пятерок,

у которых не существует двух различных пятерок с двумя одинаковыми первыми элементами), и, наконец, множество ТМ чисел, которые суть номера машин Тьюринга. ТМ является множеством всех номеров F_n , таких, что для каждого состояния s из $S - \{f\}$ существует пятерка в списке, начинающаяся парой $(0, s)$, а также пятерка, начинающаяся парой $(1, s)$, а также пятерка, начинающаяся парой (β, s) (так что область определения функции есть все множество $A \times (S - \{f\})$).

Чтобы определить Quint, мы должны представить пятерку чисел одним числом. Для достижения этой цели мы, следуя Куайну [7] и Смультану [8], представим последовательность чисел посредством «разделенного сочленения», т. е. записывая их одно за другим с промежутками между ними. Эти промежутки должны обладать тем свойством, что они позволяют разложить единственным образом «разделенное сочленение» в последовательность, которую оно представляет. Чтобы определить Quint, замечаем, что каждый элемент y пятерки удовлетворяет одному из отношений $A(y)$, $R(y)$, $L(y)$, $N(y)$, $St(y)$. Так как строка 22 не встречается ни в одном из этих элементов пятерок, мы используем 1221 в качестве промежутка. Тогда строка, не начинающаяся и не оканчивающаяся на 1221, но содержащая четыре вхождения 1221 и не содержащая вхождений 222, однозначно интерпретируется как некоторая пятерка (упорядоченная пятерка). Например, 212211111221112211112212 представляет пятерку $(2, 111, 1, 11, 2)$. Вообще мы определяем множество Quint следующим образом:

$$\begin{aligned} \text{Quint}(x) = (\exists x_1, x_2, x_3, x_4, x_5)_{\leq x} [x_1 1221 x_2 1221 x_3 1221 x_4 1221 x_5 = x \wedge \\ \wedge A(x_1) \wedge St(x_2) \wedge \sim f(x_2) \wedge A(x_3) \wedge St(x_4) \wedge \\ \wedge (R(x_5) \vee L(x_5) \vee N(x_5))]. \end{aligned}$$

Так как таблица — это просто последовательность пятерок, мы пользуемся аналогичным способом, чтобы изобразить последовательность пятерок одним числом. Теперь роль промежутка играет 12221, так как '22 — самая длинная строка из 2, которая появляется в пятерках. Остальные определения, ведущие к ТМ(x), говорят сами за себя.

$$\begin{aligned} \text{Table}(x) \equiv (12221) Bx \wedge (12221) Ex \wedge \sim (2222) Px \wedge \\ \wedge (y)_{\leq x} [(12221y12221) Px \wedge \sim (222) Py \rightarrow \text{Quint}(y)] \end{aligned}$$

$$\text{Quint}(x, y) \equiv \text{Quint}(x) \wedge \text{Table}(y) \wedge (12221x12221) Py,$$

$$\begin{aligned} \text{Quint}((x_1, x_2, x_3, x_4, x_5), y) \equiv (\exists x)_{\leq y} [\text{Quint}(x, y) \wedge x = \\ = x_1 1221 x_2 1221 x_3 1221 x_4 1221 x_5], \end{aligned}$$

$$\begin{aligned}
\text{Agree}(x, y) &\equiv (x_1, x_2)_{\leq x} [(x_1 1221 x_2 1221) Bx \wedge \\
&\quad \wedge (x_1 1221 x_2 1221) By \rightarrow x = y], \\
\text{Fn}(x) &\equiv \text{Table}(x) \wedge (y, z)_{\leq x} [\text{Quint}(y, x) \wedge \text{Quint}(z, x) \rightarrow \\
&\quad \rightarrow \text{Agree}(y, z)], \\
\text{TM}(x) &\equiv \text{Fn}(x) \wedge (y)_{\leq x} \{(\exists y_1, y_2, y_3, y_4)_{\leq x} \\
&\quad [\text{Quint}(y_1 1221 y 1221 y_2 1221 y_3 1221 y_4) \vee \\
&\quad \vee \text{Quint}(y_1 1221 y_2 1221 y_3 1221 y 1221 y_4) \wedge \\
&\quad \wedge y \neq 11] \rightarrow (\exists z_1, z_2, z_3, w_1, w_2, w_3, v_1, v_2, v_3)_{\leq x} \\
&\quad [\text{Quint}((1, y, z_1, z_2, z_3), x) \wedge \text{Quint}((2, y, w_1, w_2, w_3), x) \wedge \\
&\quad \wedge \text{Quint}((11, y, v_1, v_2, v_3), x)]\}.
\end{aligned}$$

В дополнение к номерам машин Тьюринга нам нужны номера лент и мгновенных описаний, чтобы мы могли анализировать вычисление функции на машине Тьюринга числовыми методами. Мы сопоставим число $1221\bar{a}_1 1221\bar{a}_2 \dots \bar{a}_{n-1} 1221\bar{a}_n 1221$ ленте $a_1 a_2 \dots a_{n-1} a_n$, где $a_i \in \{0, 1, \beta\}$ и $\bar{0}=1, \bar{1}=2, \bar{\beta}=11$. Тогда, чтобы сопоставить некоторое число мгновенному описанию (t, s, p) , мы заменяем сначала положительное целое p на номер P первых p символов ленты t . Например, в $(a_1 a_2 a_3 a_4, s, 3)$ мы заменяем 3 на номер $a_1 a_2 a_3$, т. е. на $1221\bar{a}_1 1221\bar{a}_2 1221\bar{a}_3 1221$. Теперь мы сопоставляем описанию (t, s, p) номер $T12221S12221P$, где T — номер ленты t и S — номер состояния s . Формально мы получаем

$$\begin{aligned}
\text{Tape}(x) &\equiv (1221) Bx \wedge (1221) Ex \wedge \sim (222) Px \wedge \\
&\quad \wedge (y)_{\leq x} [(1221 y 1221) Px \wedge \sim (22) Py \rightarrow A(y)]. \\
\text{ID}(x, y, z, w) &\equiv [x = y 12221 z 12221 w \wedge \text{Tape}(y) \wedge \\
&\quad \wedge \text{Tape}(w) \wedge w B y \wedge S t(z)]. \\
\text{ID}(x) &\equiv (\exists y, z, w)_{\leq x} (\text{ID}(x, y, z, w)).
\end{aligned}$$

В нашем определении вычисления на машине Тьюринга одно мгновенное описание приводило к другому при помощи операции $\rightarrow$, определенной в четыре этапа. Сейчас мы отобразим каждый этап в определении функции Yield (где первые три являются подслучаями первого пункта определения $\rightarrow$).

$$\begin{aligned}
\text{Yield}_{1a}(x, y, z) &\equiv \text{TM}(z) \wedge (\exists x_1, x_2, a_1, z_1)_{\leq x} (\exists a_2, z_2)_{\leq y} \\
&\quad [\text{ID}(x, x_1 a_1 x_2, z_1, x_1 a_1 1221) \wedge \\
&\quad \wedge \text{ID}(y, x_1 a_2 x_2, z_2, x_1 a_2 1221) \wedge \\
&\quad \wedge \text{Quint}((a_1, z_1, a_2, z_2, 11), z)],
\end{aligned}$$

$$\begin{aligned} \text{Yield}_{1b}(x, y, z) \equiv & \text{TM}(z) \wedge (\exists x_1, x_2, a_1, z_1)_{\leq x} (\exists a_2, a_3, z_2)_{\leq y} \\ & [\text{ID}(x, x_1 a_1 1221 a_3 x_2, z_1, x_1 a_1 1221) \wedge \\ & \wedge \text{ID}(y, x_1 a_2 1221 a_3 x_2, z_2, x_1 a_2 1221 a_3 1221) \wedge \\ & \wedge \text{Quint}((a_1, z_1, a_2, z_2, 1), z)], \end{aligned}$$

$$\begin{aligned} \text{Yield}_{1c}(x, y, z) \equiv & \text{TM}(z) \wedge (\exists x_1, x_2, a_1, a_3, z_1)_{\leq x} (\exists a_2, z_2)_{\leq y} \\ & [\text{Quint}((a_1, z_1, a_2, z_2, 2), z) \wedge \\ & \wedge \text{ID}(x, x_1 a_3 1221 a_1 1221 x_2, z_1, x_1 a_3 1221 a_1 1221) \wedge \\ & \wedge \text{ID}(y, x_1 a_3 1221 a_2 1221 x_2, z_2, x_1 a_3 1221)], \end{aligned}$$

$$\begin{aligned} \text{Yield}_2(x, y, z) \equiv & \text{TM}(z) \wedge (\exists x_1, a_1, z_1)_{\leq x} (\exists a_2, z_2)_{\leq y} \\ & [\text{Quint}((a_1, z_1, a_2, z_2, 1), z) \wedge \\ & \wedge \text{ID}(x, x_1 a_1 1221, z_1, x_1 a_1 1221) \wedge \\ & \wedge \text{ID}(y, x_1 a_2 1221 111221, z_2, x_1 a_2 1221 111221)], \end{aligned}$$

$$\begin{aligned} \text{Yield}_3(x, y, z) \equiv & \text{TM}(z) \wedge (\exists x_1, a_1, a_3, z_1)_{\leq x} (\exists a_2, z_2)_{\leq y} \\ & [\text{Quint}((a_1, z_1, a_2, z_2, 2), z) \wedge \\ & \wedge \text{ID}(x, x_1 a_3 1221 a_1 1221, z_1, x_1 a_3 1221 a_1 1221) \wedge \\ & \wedge \text{ID}(y, x_1 a_3 1221 a_2 1221, z_2, x_1 a_3 1221)], \end{aligned}$$

$$\begin{aligned} \text{Yield}_4(x, y, z) \equiv & \text{TM}(z) \wedge (\exists x_1, a_1, a_3, z_1)_{\leq x} (\exists z_2)_{\leq y} \\ & [\text{Quint}((a_1, z_1, 11, z_2, 2), z) \wedge \\ & \wedge \text{ID}(x, x_1 a_3 1221 a_1 1221, z_1, x_1 a_3 1221 a_1 1221) \wedge \\ & \wedge \text{ID}(y, x_1 a_3 1221, z_2, x_1 a_3 1221)], \end{aligned}$$

$$\begin{aligned} \text{Yield}(x, y, z) \equiv & [\text{Yield}_{1a}(x, y, z) \vee \text{Yield}_{1b}(x, y, z) \vee \\ & \vee \text{Yield}_{1c}(x, y, z) \vee \text{Yield}_2(x, y, z) \vee \\ & \vee \text{Yield}_3(x, y, z) \vee \text{Yield}_4(x, y, z)]. \end{aligned}$$

Поскольку отношение Yield принадлежит $\mathcal{E}_*^2$, следующее определение показывает, что функция Yield двух переменных также принадлежит $\mathcal{E}^2$. Смысл этой функции достаточно ясен: $\text{Yield}(x, z)$ есть гёделев номер мгновенного описания, которое непосредственно следует за мгновенным описанием с номером x на машине Тьюринга с номером z ,

$$\text{Yield}(x, z) = \min y \leq x \cdot z [\text{Yield}(x, y, z)].$$

Вычисление на машине Тьюринга заканчивается тогда, когда достигается мгновенное описание с заключительным состоянием f . Так что мы определяем $\text{Fin}(x)$ как множество чисел x , кото-

рые являются номерами мгновенных описаний, содержащих состояние f ,

$$\text{Fin}(x) \equiv (\exists y, z)_{\leq x} [\text{ID}(x, y, 11, z)].$$

Теперь мы хотим арифметизовать вычисления числовых функций, но, прежде чем мы сможем сделать это, мы должны найти метод перевода неотрицательных целых, записанных в двоичной системе счисления, в положительные целые, закодированные в двузначном коде. Точнее, нам нужна функция e из $\mathcal{E}^2$, отображающая взаимно однозначно неотрицательные целые в положительные целые. Мы фактически неявно использовали такую функцию при арифметизации лент, когда мы сопоставляли гёделев номер 1 целому 0 и гёделев номер 2 целому 1. Распространяя это сопоставление на всю двоичную запись произвольного неотрицательного целого, получаем соответствующий двузначный код некоторого положительного целого числа. Фактически эта замена 0 на 1 и 1 на 2 повсюду в двоичной записи целого n приводит к двузначному коду числа

$$n + 2^{l(n)-1} + \dots + 2^1 + 2^0 = n + 2^{l(n)} - 1.$$

Таким образом, определяя функцию e посредством $e(n) = n + 2^{l(n)} - 1$, получаем нужный перевод неотрицательных целых в положительные целые. Теперь мы должны показать, что двуместное отношение ε принадлежит $\mathcal{E}_*^2$, где $\varepsilon(x, y) \equiv e(x) = y$.

Чтобы завершить это, мы определяем сначала в $\mathcal{E}^2$ функцию остатка r (как это было сделано в [3]), функцию t , значение которой равно удвоенному первому аргументу, если второй аргумент есть 0, и в точности первому аргументу в противном случае, и, наконец, функцию $2^{l(x)}$. Тогда, так как $\mathcal{E}^2$ замкнут относительно сложения и операции урезанной разности $\div$ (см. [3]), мы покажем, что функция e принадлежит $\mathcal{E}^2$, и по теореме 1 $\varepsilon \in \mathcal{E}_*^2$. r, t и $2^{l(x)}$ определяются следующим образом:

$$\left. \begin{aligned} r(0, y) &= 0 \\ r(s(x), y) &= \sigma[s(r(x, y)), 1 \div (y \div s(r(x, y)))] \\ r(x, y) &\leq y \end{aligned} \right\} \begin{aligned} &r(x, y) \text{ есть остаток от деления } x \\ &\text{на } y, \end{aligned}$$

$$\left. \begin{aligned} t(x, 0) &= 2 \cdot x \\ t(x, s(y)) &= x \\ t(x, y) &\leq 2 \cdot x \end{aligned} \right\} \begin{aligned} &t(x, y) = 2 \cdot x, \text{ если } y = 0, \text{ и } x \text{ в противном} \\ &\text{случае,} \end{aligned}$$

$$2^{l(0)} = 2,$$

$$2^{l(s(x))} = t[2^{l(x)}, r(s(x), 2^{l(x)})],$$

$$2^{l(x)} \leq 2 \cdot (x + 1).$$

Теперь, когда мы показали, что отношение e , которое выполняется для x и y тогда и только тогда, когда $e(x) = y$, принадлежит $\mathcal{E}_*^2$ мы готовы к анализу вычислений числовых функций. Такое вычисление определяется последовательностью мгновенных описаний, в которой начальная лента имеет вид $\bar{x}_1\beta\bar{x}_2\beta \dots \bar{x}_{n-1}\beta\bar{x}_n$ и заключительная лента имеет вид $\bar{x}_0$, где $\bar{x}_i$ есть двоичное представление числа x_i . Мы начинаем с определения отношения Code , где $\text{Code}(y, x)$ имеет место тогда и только тогда, когда y есть гёделев номер ленты $\bar{x}$. Из этого отношения мы сможем затем определить функцию Decode , которая дает число (если таковое имеется), представленное на ленте в заключительном мгновенном описании, и для каждого n отношение Encode_n , где $\text{Encode}_n(x_1, \dots, x_n, y)$ имеет место тогда и только тогда, когда y — гёделев номер ленты $\bar{x}_1\beta \dots \bar{x}_{n-1}\beta\bar{x}_n$. Наконец, при помощи Encode_n мы затем определим функцию Init_n , которая дает гёделев номер начального мгновенного описания, лента которого содержит двоичную запись n аргументов Init_n , взятых в том же порядке. Из этих определений мы выводим

$$\begin{aligned} \text{Code}(y, x) \equiv & \text{Tape}(y) \wedge (\exists u)_{\leq y} \{e(x) = u \wedge (z, w)_{\leq y} \\ & \{[zBu \wedge wEz \wedge (w = 1 \vee w = 2)] \rightarrow \\ & \rightarrow (\exists v)_{\leq y} [vBy \wedge l(v) = 5 \cdot l(z) \wedge (1221w)Ev]\} \}, \end{aligned}$$

$$\text{Decode}(x) = \min y \leq x \{ \text{ID}(x) \wedge (\exists z)_{\leq x} [(z12221)Bx \wedge \text{Code}(z, y)] \},$$

$$\begin{aligned} \text{Encode}_n(x_1, \dots, x_n, x) \equiv & (\exists y_1, \dots, y_n)_{\leq x} [\text{Code}(y_1, x_1) \wedge \dots \\ & \dots \wedge \text{Code}(y_n, x_n) \wedge \\ & \wedge x = y_111y_2 \dots y_{n-1}11y_n], \end{aligned}$$

$$\begin{aligned} \text{Init}_n(x_1, \dots, x_n) = \min w \leq & 2^{6 \cdot n + 19} \cdot [(x_1 + 1) \cdot \dots \cdot (x_n + 1)]^5 \\ & \{(\exists x)_{\leq w} [\text{Encode}_n(x_1, \dots, x_n, x) \wedge \\ & \wedge (\exists y)_{\leq x} [l(y) = 9 \wedge yBx \wedge \\ & \wedge w = x12221112221y]]\}. \end{aligned}$$

Граница для w определяется из условий $l(w) = l(x) + 20$ и $l(x) = 5 \cdot (l(x_1) + \dots + l(x_n)) + 4 + 6 \cdot (n - 1)$, так что

$$\begin{aligned} w < 2 \cdot 2^{l(w)} &= 2 \cdot 2^{5 \cdot (l(x_1) + \dots + l(x_n)) + 18 + 6 \cdot n} = \\ &= 2^{6 \cdot n + 19} \cdot (2^{l(x_1)} \cdot \dots \cdot 2^{l(x_n)})^5 \leq \\ &\leq 2^{6 \cdot n + 19} \cdot [(x_1 + 1) \cdot \dots \cdot (x_n + 1)]^5. \end{aligned}$$

ЛИТЕРАТУРА

1. Davis M., Computability and unsolvability, McGraw-Hill, New York, 1958
2. Elgot C. C., Decision problems of finite automata design and related arithmetics, *Trans Amer. Math. Soc.*, **98** (1961), 21—51.
3. Grzegorzczuk A., Some classes of recursive functions, *Rozprawy Matematyczne*, Warszawa, 1953 (Русский перевод см. стр. 9—49.)
4. Kleene S. C., Introduction to metamathematics, Van Nostrand, Princeton, N. J., 1952. (Русский перевод: Клини С. К., Введение в метаматематику, ИЛ, М., 1957.)
5. Myhill J., Linear bounded automata, WADD Technical Note, 60—165, Univ Pennsylvania Report, Nr. 60-22, June 1960.
6. Péter Rózsa, Rekursive Funktionen, Akadémiai Kiadó, Budapest, 1957, 76—86. (Русский перевод: Петер Р., Рекурсивные функции, ИЛ, М., 1954.)
7. Quine W. V., Concatenation as a basis for arithmetic, *J. Symb. Logic*, **11** (1946), 105—114.
8. Smullyan R. M., Theory of formal systems, *Annals of Mathematics Studies*, № 47, Princeton Univ. Press, Princeton, N. J., 1961.
9. Tarski A., A decision method for elementary algebra and geometry, Project RAND Report R-109, The RAND Corporation, Santa Monica, Calif., 1957.

ИЕРАРХИЯ ПРИМИТИВНО РЕКУРСИВНЫХ ФУНКЦИЙ ¹⁾

Дж. П. Клив

1. ВВЕДЕНИЕ

Гжегорчик [1] построил цепь типа ω $\mathcal{E}^0 \subset \mathcal{E}^1 \subset \mathcal{E}^2 \subset \dots$ классов рекурсивных функций, где $\mathcal{E}^3$ есть множество K элементарных по Кальмару функций и $\bigcup_{n=0}^{\infty} \mathcal{E}^n$ является множеством P примитивно рекурсивных функций. Ричи [2] построил цепь типа ω $F^0 \subset F^1 \subset F^2 \subset \dots$ классов функций, так что F^0 есть множество функций, вычислимых на конечных автоматах, F^{i+1} есть класс функций, вычислимых на машинах Тьюринга, которые используют при вычислении ленту с длиной, являющейся функцией (тех же аргументов) из F^i , и $\bigcup_{n=0}^{\infty} F^n = K$. Таким образом, у Ричи сложность вычисления измеряется длиной ленты, необходимой для его выполнения.

В этой статье строится ω^2 -цепь классов функций, вычислимых на одной разновидности вычислительных машин. Получение свойств этих функций зависит, однако, от эквивалентного определения, данного с помощью простого типа рекурсии (теорема 1). Вместо машины Тьюринга оказалось удобным использовать «неограниченные машины с регистрами» ²⁾ (Шепердсон и Стургис [3]), известные как « J -ограниченные машины». Этот метод обладает тем преимуществом, что, во-первых, он ближе к реальной вычислительной практике, так что можно свободно использовать такие понятия, как «код команды», «диаграмма переходов», «программа» и т. д., и в то же время удастся избежать произвольных представлений чисел в машинах Тьюринга.

Если дано множество Σ функций, то можно определить код команды для J -ограниченной машины и, следовательно, построить « Σ -программу». Понятие «шага» в вычислении есть понятие относительно Σ . Если определить сложность вычисления как число шагов, необходимых для вычисления, то метод Ричи построения классов функций дает цепь типа ω $E(\Sigma)_0 \subseteq E(\Sigma)_1 \subseteq E(\Sigma)_2 \subseteq \dots$ классов со свойствами, подобными свой-

¹⁾ Cleave J. P., A hierarchy of primitive recursive functions, *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 9, № 4 (1963), 331—346.

²⁾ В оригинале «unlimited register machine». — Прим. перев.

ствам последовательности $F^0, F^1, F^2, \dots$. Итак, пусть $E(\Sigma)_{-1}$ — множество константных функций, $E(\Sigma)_i$ — множество функций, вычислимых на J -ограниченных машинах при помощи Σ -программ и таких, что число шагов, необходимое для вычислений, есть функция (тех же аргументов) из $E(\Sigma)_{i-1}$. Следовательно, каждому множеству Σ функций соответствует цепь типа ω^2 $E_0 \subseteq E_1 \subseteq E_2 \subseteq \dots \subseteq E_{\omega r+s} \dots$ классов рекурсивных функций, где

$$\left. \begin{aligned} E_0 &= E(\Sigma)_0, \\ E_{\omega r+s} &= E(E_{\omega r})_s, \\ E_{\omega r} &= \bigcup_{i < \omega r} E_i, \\ E_{\omega^2} &= \bigcup_{i < \omega^2} E_i, \end{aligned} \right\} r, s < \omega. \quad (1)$$

Итерация функций, роль которой в построении примитивно рекурсивных функций была изучена Р. М. Робинсоном [4], занимает естественное место в получении свойств классов $\{E_r\}$. Так, каждый класс $E_{\omega(r+1)}$ есть замыкание относительно композиции множества $F(E_{\omega r})$ функций, определенных из $E_{\omega r}$ совместной итерацией (теорема 2.1.2). Некоторые замечательные свойства этих классов устанавливаются ниже в теореме 2.

Особый интерес представляет случай $\Sigma = \{+, \times, \delta\}$ ($\delta(x, y) = 0, 1$ в соответствии с $x \neq y$ или $x = y$). Тогда $E_{\omega s} = \mathcal{E}^{s+2}$ для $s \geq 1$, следовательно, $K = E_\omega$, $P = E_{\omega^2}$.

2. СЛОЖНОСТЬ ВЫЧИСЛЕНИЙ НА J -ОГРАНИЧЕННЫХ МАШИНАХ

Неограниченная машина с регистрами (URM) ранга r состоит из r регистров $R_1, R_2, \dots, R_r$, каждый из которых может содержать любое натуральное число. Содержимое регистров, также обозначаемое через $R_1, R_2, \dots, R_r$, может быть изменено программой; символы " R_1 ", " R_2 ", $\dots$, " R_r " появляются как адреса в командах.

Пусть $\mathbf{N}$ обозначает множество натуральных чисел. Пусть далее Σ — счетное множество функций, определенных на $\mathbf{N}$ со значениями в $\mathbf{N}$, и пусть $\Sigma^{[n, 1]}$ обозначает подмножество функций из Σ от n переменных. Если Σ содержит $+$, $\times$, δ , то Σ -команда для URM есть элемент множества $\mathfrak{Z}(\Sigma) \cup \mathfrak{B}$, где

1) $\mathfrak{Z}(\Sigma)$ — множество Σ -арифметических команд, построенное из Σ следующим образом. Каждой $F \in \Sigma^{[n, 1]}$ соответствует множество $\mathfrak{Z}(F)$ команд

$$F(R_{\alpha_1}, R_{\alpha_2}, \dots, R_{\alpha_n}) \rightarrow R_\beta, \quad \alpha_1, \alpha_2, \dots, \alpha_n, \beta = 1, 2, \dots,$$

т. е. „поместить $F(R_{\alpha_1}, R_{\alpha_2}, \dots, R_{\alpha_n})$ в R_β , уничтожив предыдущее содержимое R_β “. Тогда

$$\mathfrak{Z}(\Sigma) = \bigcup_{F \in \Sigma} \mathfrak{Z}(F).$$

2) $\mathfrak{B}$ есть множество $\{J_i(\alpha, \beta)\}$, $i, \alpha, \beta = 1, 2, \dots$, **команд выбора**¹⁾ для распознавания содержимого R_i , $i = 1, 2, \dots$.

Конечная последовательность $\{I(1), I(2), \dots, I(n)\}$ членов $\mathfrak{Z}(\Sigma) \cup \mathfrak{B}$ называется Σ -**программой**, если

а) $I(n) \in \mathfrak{B}$,

б) из $I(m) = J_i(\alpha, \beta)$, $1 \leq m \leq n$, следует $\alpha \leq n$ и $\beta \leq n$.

$\mathfrak{A}(\Sigma)$ обозначает множество всех Σ -программ.

Ранг $r(I)$ Σ -команды I определяется как

$$r(I) = \begin{cases} \max(\alpha_1, \alpha_2, \dots, \alpha_n, \beta), & \text{если } I = F(R_{\alpha_1}, \dots, R_{\alpha_n}) \rightarrow R_\beta, \\ \max(i, \alpha, \beta), & \text{если } I = J_i(\alpha, \beta). \end{cases}$$

Ранг $r(P)$ последовательности $P = \{I(1), \dots, I(n)\}$ команд есть

$$r(P) = \max(r(I(1)), \dots, r(I(n))).$$

Команды Σ -программы $P = \{I(1), \dots, I(n)\}$ выполняются последовательно на URM ранга $\geq r(P)$: сначала $I(1)$, затем $I(2), \dots$, пока не встретится команда выбора, скажем $I(m) = J_i(\alpha, \beta)$. $I(m)$ решает, которая из двух команд будет выполняться следующей — $I(\alpha)$, если $R_i = 1$, либо $I(\beta)$, если $R_i \neq 1$. $I(m)$ есть **динамический стоп** в P , если $I(m) = J_i(m, m)$. Коль скоро такой стоп достигается в процессе вычисления, программа продолжает выполнять $I(m)$ согласно P , однако содержимое регистров остается неизменным. Заметим, что программа вовсе не обязана иметь динамический стоп.

Пусть P есть Σ -программа ранга $\geq n$ и $D_n(P)$ — множество $(n+1)$ -ок $(x_1, \dots, x_n, y)$, таких, что если P начинает работать с $R_i = x_i$ для $1 \leq i \leq n$ и $R_i' = 0$, $i > n$, то динамический стоп достигается с $R_1 = y$. Тогда P вычисляет (частичную) функцию f n переменных тогда и только тогда, когда $D_n(P)$ является графиком f . Можно показать, что f является частично рекурсивной функцией тогда и только тогда, когда существует Σ -программа, вычисляющая ее на URM.

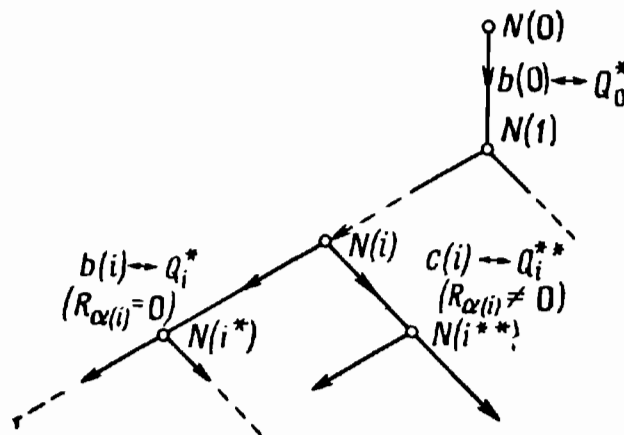
Более узкий класс функций получается, если снабдить URM устройством, которое гарантирует нам, что вычисления всегда

¹⁾ В оригинале «jump orders». — Прим. перев.

имеют конечную длину. Таким образом, *J-ограниченная машина* (JM) ранга r есть URM ранга r вместе со специальным регистром J , на который не распространяется действие арифметических команд; Σ -программы, определенные точно так же, как и для URM, подчиняются следующим ограничениям, налагаемым на них регистром J .

1. Никакая команда не может выполняться, если $J=0$.

2. После выполнения команды выбора из J вычитается 1. Следовательно, если Σ -программа начинает работать на JM с $n > 0$ в J , то машина останавливается (но это не динамический стоп!) с 0 в J сразу после выполнения в процессе вычисления



Р и с. 1.

n -й команды выбора. Число в J поэтому определяет количество двуисходных¹⁾ решений, выполняемых в процессе вычисления на Σ -программе.

Σ -программе $P = \{I(1), \dots, I(k)\}$ естественным образом соответствует ориентированный граф $G(P)$ — **диаграмма переходов** P , вершины которого соответствуют командам выбора, а ребра — последовательностям арифметических команд, встречающихся в процессе вычисления с помощью P между двумя последовательными командами выбора (Петер [5]).

Пусть $r = r(P)$; предположим, что P имеет команды выбора $I(j_m) = J_{\alpha(j_m)}(p(j_m), q(j_m))$, $1 \leq m \leq s$, где $j_1 < j_2 < \dots < j_s = k$. Положим $0^* = 1$ и $i^* = (\mu m)(j_m \geq p(i))$, $i^{**} = (\mu m)(j_m \geq q(i))$ для $1 \leq m \leq s$. Тогда $G(P)$ (рис. 1) состоит из вершины $N(0)$, вершин $N(1), \dots, N(s)$, соответствующих $I(j_1), \dots, I(j_s)$, и ребер $b(0), b(i), c(i)$, $1 \leq i \leq s$, где

- 1) $N(i), N(i^*)$ — начальная и конечная вершины $b(i)$, $1 \leq i \leq s$;
- 2) $N(i), N(i^{**})$ — начальная и конечная вершины $c(i)$, $1 \leq i \leq s$;

¹⁾ В оригинале «binary decision». — Прим. перев.

3) $b(i)$ соответствует последовательность Q_i^* арифметических команд:

$$Q_i^* = \begin{cases} \{I(p(i)), \dots, I(j_{i*} - 1)\}, & \text{если } p(i) < j_{i*} \text{ \& } 1 \leq i \leq s, \\ \{\emptyset\} \text{ (пустая последовательность),} & \text{если} \\ \quad [p(i) = j_{i*} \text{ \& } 1 \leq i \leq s] \vee [i = 0 \text{ \& } j_1 = 1], \\ \{I(1), \dots, I(j_1 - 1)\}, & \text{если } i = 0 \text{ \& } j_1 > 1; \end{cases}$$

4) $c(i)$ соответствует последовательность Q_i^{**} :

$$Q_i^{**} = \begin{cases} \{I(q(i)), \dots, I(j_{i**} - 1)\}, & \text{если } q(i) < j_{i**} \text{ \& } 1 \leq i \leq s, \\ \{\emptyset\}, & \text{если } [q(i) = j_{i**} \text{ \& } 1 \leq i \leq s]. \end{cases}$$

Вычислению с помощью P соответствует путь в $G(P)$, начинающийся в $N(0)$. Если путь достигает вершины $N(i)$, соответствующей $J_{\alpha(i)}(p(i), q(i))$, то далее он идет по ребру $b(i)$, если $R_{\alpha(i)} = 1$, и по $c(i)$, если $R_{\alpha(i)} \neq 1$. Если P начинает работать на JM с $J = n$, то вычисление, начинающееся в $N(0)$, образует путь в $G(P)$ длины n : Σ -сложность вычисления определяется числом n , т. е. числом двуисходных решений (выборов), производимых в процессе вычисления.

Пусть h есть функция на $\mathbf{N}^n$. Говорят, что функция f , определенная на $\mathbf{N}^n$, (h, Σ) -вычислима, если существуют число j и Σ -программа P , такие, что, начав работать с $J = h(x_1, \dots, x_n)$, $R_i = x_i$ для $1 \leq i \leq n$, $R_i = 0$ для $n < i \leq m$, JM ранга $m \geq \max(j, r(P))$ останавливается с $R_j = f(x_1, \dots, x_n)$. Мы пишем $f = [P, h]$; f также определена на $\mathbf{N}^n$.

3. КЛАССЫ ФУНКЦИЙ, ВЫЧИСЛИМЫХ НА J -ОГРАНИЧЕННЫХ МАШИНАХ

Если использовать понятие «сложность вычисления», введенное в разд. 2, то метод Ричи построения классов вычислимых функций дает последовательность $E(\Sigma)_0, E(\Sigma)_1, E(\Sigma)_2, \dots, E(\Sigma)_\omega$, определенную посредством

$$\left. \begin{aligned} E(\Sigma)_0 &= \{f \mid f = [P, h], P \in \mathfrak{A}(\Sigma), h \in C\}, \\ E(\Sigma)_{r+1} &= \{f \mid f = [P, h], P \in \mathfrak{A}(\Sigma), h \in E(\Sigma)_r\}, \\ E(\Sigma)_\omega &= \bigcup_{r < \omega} E(\Sigma)_r, \end{aligned} \right\} \quad (2)$$

где C является множеством константных функций. Легко проверить, что $r < s$ влечет $E(\Sigma)_r \subseteq E(\Sigma)_s$.

Цепь $\{E(\Sigma)_r\}$ классов может быть охарактеризована также рекурсиями. Пусть $U_i^r(x_1, \dots, x_r) = x_i$, если $r \geq i$, и 0 в противном случае; $\Sigma^* = \Sigma \cup \{U_i^r\}$ и $Cl(\Sigma^*)$ — замыкание Σ^* относительно композиции. Пусть также $A_i \in Cl(\Sigma^*)^{[n, 1]}$, $m \leq n$. Рассмотрим рекурсии (совместные итерации):

$$\left. \begin{aligned} R_i(x_1, \dots, x_m, 0) &= x_i, \text{ если } 1 \leq i \leq m, 0, \text{ если } m < i \leq n, \\ R_i(x_1, \dots, x_m, y+1) &= A_i(R_1(x_1, \dots, x_m, y), \dots, R_n(x_1, \dots, x_m, y)). \end{aligned} \right\} \quad (3)$$

$F(\Sigma)$ обозначает совокупность функций R , определенных совместной итерацией из $Cl(\Sigma^*)$.

Теорема 1. *Функция f , определенная на $\mathbf{N}^n$, является (h, Σ) -вычислимой тогда и только тогда, когда существует функция $R \in F(\Sigma)^{[n+1, 1]}$, такая, что $f(x_1, \dots, x_n) \equiv R(x_1, \dots, x_n, h(x_1, \dots, x_n))$.*

Здесь ω -цепь $\{E(\Sigma)_r\}$ можно продолжить за ω до ω^2 , используя процесс, определенный в (1). Это равносильно построению для каждого предельного трансфинитного числа ωr нового множества $\mathfrak{S}(E_{\omega r})$ арифметических команд, которое затем используется для вычисления функций в $E_{\omega r+s}$, $1 \leq s \leq \omega$. То, что построение стягивается к ω^2 , ясно из следующего соображения. Пусть $P = \{I(1), \dots, I(k)\}$ есть E_{ω^2} -программа. Команды $I(r)$, $1 \leq r \leq k$, построены из функций $F_r \in E_{\omega^2}$. Однако если $F_r \in E_{\omega^2}$, то существует число a_r , такое, что $F_r \in E_{\omega a_r}$, $1 \leq r \leq k$. Если $z = \max(a_1, \dots, a_k)$, то P является $E_{\omega z}$ -программой. Следовательно, построение (2) с $\Sigma = E_{\omega z}$ дает цепь, уже содержащуюся в $\{E_r\}$, $r < \omega^2$.

Свойства ω^2 -цепи $\{E_r\}$, $r < \omega^2$, определяются, конечно, множеством Σ , из которого строится исходное множество арифметических команд. Особый интерес представляют „нормальные“ множества. Σ является **нормальным**, если $Cl(\Sigma^*)$ содержит строго возрастающую функцию $h(y)$, такую, что для всякой функции $g \in Cl(\Sigma^*)^{[n, 1]}$ существует целое m и что $f(m, \max(x_i)) > g(x_1, \dots, x_n)$ для всех x_i , где f определяется рекурсией $f(0, y) = y + 1$, $f(n+1, y) = h(f(n, y))$. Если $\Sigma = \{+, \times, \delta\}$, то Σ нормально с $h(y) = (y+1)^2$.

Теорема 2¹⁾.

1.1. Пусть $a < \omega$ и $f \in E_{\omega a+r}^{[n, 1]}$, $g_i \in E_{\omega a+s}^{[m, 1]}$ для $1 \leq i \leq n$; тогда если $F(x_1, \dots, x_m) \equiv f(g_1(x_1, \dots, x_m), \dots, g_n(x_1, \dots, x_m))$, то $F \in E_{\omega a+r+s}^{[m, 1]}$.

¹⁾ В этой теореме решается вопрос о том, что дают различные примитивно рекурсивные операции в применении к функциям класса E_a ($a < \omega^2$). Ср. леммы 1, 2, 4 из статьи Ричи в этом сборнике. — *Прим. ред.*

1.2. Если $a < \omega$, то $F(E_{\omega a}) \subseteq E_{\omega(a+1)}$ и $Cl(F(E_{\omega a})) = E_{\omega(a+1)}$.

2. Если Σ — нормальное множество функций, то $E_r \subset E_s$, $r < s < \omega^2$.

3.1. Если Σ — нормальное множество функций, $a < \omega$, $h \in E_{\omega a}^{[n+2, 1]}$, $g \in E_{\omega a+r}^{[n, 1]}$ и

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, y+1) &= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)), \end{aligned}$$

то $f \in E_m^{[n+1, 1]}$, $m = \omega a + \max(r, 1)$.

3.2. Если Σ — нормальное множество функций, $a < \omega$, $h \in E_{\omega a+s}^{[n+2, 1]}$, $s > 0$, $g \in E_{\omega a+r}^{[n, 1]}$ и существует функция $k \in E_{\omega a+t}^{[n+1, 1]}$, такая, что $f(x_1, \dots, x_n, y) \leq k(x_1, \dots, x_n, y)$, где

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, y+1) &= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)), \end{aligned}$$

тогда $f \in E_m^{[n+1, 1]}$, где $m = \omega a + \max(r, s+t)$.

3.3. Если Σ — нормальное множество, тогда для всех r , $1 \leq r \leq \omega^2$, E_r замкнуто относительно операций ограниченного суммирования и умножения, т. е. если $F \in E_r^{[n+1, 1]}$, то $f, g \in E_r^{[n+1, 1]}$, где

$$\begin{aligned} f(x_1, \dots, x_n, y) &= \sum_{i < y} F(x_1, \dots, x_n, i), \\ g(x_1, \dots, x_n, y) &= \prod_{i < y} F(x_1, \dots, x_n, i). \end{aligned}$$

Рекурсивные равенства, определяющие работу J -ограниченных машин, приводят естественным образом к получению классов E_r с помощью совместной итерации, с другой стороны, ограниченная рекурсия играет решающую роль в порождении классов $\mathcal{E}^r$ Гжегорчика. Так, $\mathcal{E}^n$ определяется как наименьший класс функций, содержащий $x+1$, $U_1^2(x, y) = x$, $U_2^2(x, y) = y$, $f_n(x, y)$ и замкнутый относительно подстановки и ограниченной рекурсии, причем

$$\left. \begin{aligned} f_0(x, y) &= y+1, \\ f_1(x, y) &= x+y, \\ f_2(x, y) &= (x+1)(y+1), \\ \text{для } n \geq 2 \quad \left\{ \begin{aligned} f_{n+1}(0, y) &= f_n(y+1, y+1), \\ f_{n+1}(x+1, y) &= f_{n+1}(x, f_{n+1}(x, y)). \end{aligned} \right. \end{aligned} \right\} \quad (4)$$

Соотношение между этими двумя последовательностями классов устанавливает

Теорема 3. Если $\Sigma = \{+, \times, \delta\}$, тогда для $s \geq 1$ $E_{\omega s} = \mathcal{E}^{s+2}$.

4. ПРЕДСТАВЛЕНИЕ ПРОГРАММ СОВМЕСТНЫМИ РЕКУРСИЯМИ

4.1. Обозначения. Согласно определению функции, вычислимой на JM, результат вычисления берется из наперед заданного регистра после того, как программа остановится. Естественно расширить определение до функций со значениями в $\mathbf{N}^n$ (для всякого n), беря результаты вычислений из наперед заданной n -ки регистров. Следующие определения устанавливают обозначения для функций из $\mathbf{N}^m$ в $\mathbf{N}^n$, которые удобны для выражения свойств функций, вычислимых при помощи Σ -программ.

1. Если A обозначает n -ку $(a_1, a_2, \dots, a_n)$ объектов (функций или чисел), тогда $(A)_r$ обозначает r -ю компоненту A , если $r \leq n$, и 0, если $r > n$. Если не возникает недоразумений, мы пишем A_r вместо $(A)_r$.

2. Пусть S — множество функций натурального аргумента. Тогда $S^{[m, 1]}$ обозначает подмножество функций m переменных в S . $S^{[m, n]}$ обозначает множество n -ок $(A_1, A_2, \dots, A_n)$, где $A_i \in S^{[m, 1]}$, $1 \leq i \leq n$. Таким образом, если $A \in S^{[m, n]}$, то A отображает $\mathbf{N}^m$ в $\mathbf{N}^n$. Пусть $\mathbf{M}^{[m, n]}$ обозначает множество всех функций на $\mathbf{N}^m$ со значениями в $\mathbf{N}^n$.

3. Если $A \in S^{[m, n]}$, $B \in S^{[l, m]}$, то AB определяется посредством $(AB)_i(x) = A_i(B_1(x), \dots, B_m(x))$ для всех $x \in \mathbf{N}^l$ и $1 \leq i \leq n$. Если S замкнуто относительно композиции, то $AB \in S^{[l, n]}$. I_m^n обозначает $(U_1^n, U_2^n, \dots, U_m^n)$.

4. С Σ -программой P ранга r и $h \in \mathbf{M}^{[s, 1]}$ ($s \leq r$) свяжем функцию $\pi(P, h) \in \mathbf{M}^{[r, r]}$, такую, что для всех $X \in \mathbf{N}^s$ если P работает на JM ранга r с $(R_1, \dots, R_r) = I_r^s$ и $J = h(X)$, то P останавливается с $(R_1, \dots, R_r) = \pi(P, h) I_r^s(X)$.

Говорят, что функция $A \in \mathbf{M}^{[n, m]}$ (h, Σ)-вычислима, $h \in \mathbf{M}^{[n, 1]}$, если существуют числа $\alpha(1), \dots, \alpha(m)$ и программа P , такие, что $r = r(P) \geq \max_{1 \leq s \leq m} (\alpha(s))$ и $A_i = (\pi(P, h I_r^n) I_r^n)_{\alpha(i)}$, $1 \leq i \leq m$.

Для $m = 1$ это определение совпадает с прежним определением на стр. 98.

Определение классов $E(\Sigma)_r$ в (2) может быть теперь расширено: $E(\Sigma)_{r+1}^{[n, m]}$ есть множество функций $A \in \mathbf{M}^{[n, m]}$, таких, что

существует $h \in E(\Sigma)_r^{[n, 1]}$, для которой A является (h, Σ) -вычислимой.

Заметим, что если $A \in E(\Sigma)_r^{[n, m]}$, то $A_i \in E(\Sigma)_r^{[n, 1]}$. Однако нужно еще доказать, что если $f_1, \dots, f_m \in E(\Sigma)_r^{[n, 1]}$, то $(f_1, \dots, f_m) \in E(\Sigma)_r^{[n, m]}$ (следствие 3).

5. Пусть $F, G \in \mathbf{M}^{[n, 1]}$. Мы пишем $F \geq G$, если для всех $X \in \mathbf{N}^n$ $F(X) \geq G(X)$, и определяем $F + G$ посредством $(F + G)(X) = F(X) + G(X)$. Определим $\max(F, G)$ как $\max(F, G)(X) \equiv \max(F(X), G(X))$.

6. Если $A \in \mathbf{M}^{[n, n]}$, то A^x определяется рекурсивно: $A^0 = I_n^n$, $A^{x+1} = AA^x$. Это дает нам краткое обозначение для совместных рекурсий (3). Так как из $A_i \in Cl(\Sigma^*)^{[n, 1]}$ имеем $A \equiv (A_1, \dots, A_n) \in Cl(\Sigma^*)^{[n, n]}$, то, следовательно, и для $X \in \mathbf{N}^m$ имеем

$$(R_1, \dots, R_n)(X, y) = A^y I_n^m(X).$$

Таким образом, $R_i \in F(\Sigma)^{[m+1, 1]}$ тогда и только тогда, когда существуют $A \in Cl(\Sigma^*)^{[n, n]}$, $n \geq m$ и $i \leq n$, такие, что $R_i(X, y) \equiv (A^y I_n^m(X))_i$.

7. Нам будут полезны функции J, K, L , нумерующие пары ¹⁾:

$$J(x, y) = \frac{(x+y)(x+y+1)}{2} + x,$$

$$K(z) = z \div \frac{1}{2} \left[\frac{[\sqrt{8z+1}] + 1}{2} \right] \cdot \left[\frac{[\sqrt{8z+1}] - 1}{2} \right],$$

$$L(z) = \left[\frac{[\sqrt{8z+1}] + 1}{2} \right] \div (K(z) + 1).$$

Они связаны тождествами $J(Kz, Lz) \equiv z$, $KJ(x, y) \equiv x$, $LJ(x, y) \equiv y$. Далее мы будем пользоваться функцией β_m , такой, что для $X \in \mathbf{N}^m$

$$\beta_m(X) = J(X_1, (\dots J(X_{m-2}, J(X_{m-1}, X_m)) \dots)).$$

Если Σ содержит $\{+, \times, \delta\}$, то

$$J \in E(\Sigma)_1^{[2, 1]^2},$$

$$K, L \in E(\Sigma)_1^{[1, 1]}, \quad \beta_m \in E(\Sigma)_1^{[m+1, 1]}.$$

¹⁾ В оригинале $J(x, y) = (x+y)^2 + x$, $Kz = z \div [\sqrt{z}]^2$, $Lz = [\sqrt{z}] \div Kz$. Однако эти функции не дают взаимно однозначного отображения N^2 на N . — Прим. перев.

²⁾ В оригинале $J \in E(\Sigma)_0^{[2, 1]}$. — Прим. перев.

4.2. Представление программ. С командой $I = f(R_{\alpha_1}, \dots, R_{\alpha_n}) \rightarrow R_\beta$ и числом $m \geq r(I)$ свяжем функцию $q_m(I) \in \Sigma^{*[m, m]}$, где для $X \in \mathbf{N}^m$

$$(q_m(I)(X))_i = \begin{cases} X_i, & \text{если } 1 \leq i \leq m \text{ и } i \neq \beta, \\ f(X_{\alpha_1}, \dots, X_{\alpha_n}), & \text{если } i = \beta. \end{cases}$$

Если команда I выполняется на URM U ранга m с $R_i = X_i$, $1 \leq i \leq m$, то после окончания ее выполнения $(q_m(I)(X))_i$ будет помещено в R_i . Следовательно, если $C = \{I(1), \dots, I(k)\}$ является последовательностью Σ -арифметических команд ранга m , то после окончания работы C на U с $R_i = X_i$, $1 \leq i \leq m$, $(q_m(C)(X_1, \dots, X_m))_i$ будет помещено в R_i , где

$$q_m(C) = q_m(I(k)) q_m(I(k-1)) \dots q_m(I(1)).$$

Ясно, что $q_m(C) \in Cl(\Sigma)^{*[m, m]}$.

Если C есть нулевая последовательность команд, то полагаем $q_m(C) = I_m^m$.

Пусть $A \in Cl(\Sigma^*)^{(m, m)}$, тогда, вообще говоря, невозможно найти последовательность C Σ -арифметических команд, использующих только $R_1, \dots, R_m$, такую, что $q_m(C) = A$. Тем не менее, используя дополнительные регистры как «рабочее пространство», A можно вычислить в следующем смысле: существует последовательность C (ранга $n \geq m$) Σ -арифметических команд, такая, что $(q_n(C)I_n^m)_i = A_i$, $1 \leq i \leq m$. Здесь и далее $\Sigma \ni \{ \times, +, \delta \}$; $0 \in Cl(\times, +, \delta)$, где $0(x) \equiv 0^1$. Следовательно, $0 \in Cl(\Sigma^*)^{[1, 1]}$. Отсюда следует, что для каждой $B \in Cl(\Sigma^*)^{[m, m']}$ можно построить такую последовательность C ранга $n \geq \max(m, m')$ Σ -арифметических команд, что

$$(q_n(C)I_n^m)_i = \begin{cases} B_i, & 1 \leq i \leq m', \\ OU_i^m, & m' < i \leq n. \end{cases} \quad (5)$$

Таким образом, если C работает на URM ранга n и $R_i = x_i$ для $1 \leq i \leq m$, $R_i = 0$ для $i > m$, то в регистрах R_i , $1 \leq i \leq m$, будут помещены $(B(x))_i$, в остальных — нули. Отсюда сразу следует лемма 1.

Лемма 1. Пусть $h \in \mathbf{M}^{[1, 1]}$, $A \in Cl(\Sigma^*)^{[m, m]}$. Тогда если $D(X) \equiv A^{h(X)}I_m^l(X)$, то D (h, Σ) -вычислима.

Обращение леммы 1 и дальнейшие следствия устанавливают соотношение между понятиями « Σ -программа» и «совместная Σ -итерация».

¹⁾ $0(x) = \delta(x, x + \delta(x, x))$. — Прим. ред.

Лемма 2. Пусть $P = \{I(1), \dots, I(k)\}$ есть Σ -программа ранга r , имеющая команды выбора $I(j_1), \dots, I(j_s)$. Тогда существует такая функция $A \in Cl(\Sigma^*)^{[r+1, r+1]}$, что если P начинает работать на URM с $R_i = X_i$, $1 \leq i \leq r$, $X \in \mathbf{N}^n$, то при m -й команде выбора $I(j_p)$, встречающейся в вычислении при помощи P ,

$$(A^m I_{r+1}^n(X))_i = \begin{cases} R_i, & \text{если } 1 \leq i \leq r, \\ p, & \text{если } i = r+1. \end{cases}$$

Доказательство. Заметим сначала, что $1, z, \bar{z} \in Cl(\{\times, +, \delta\})$, где ¹⁾

$$1(x) \equiv 1, \quad z(x) = \begin{cases} 1, & \text{если } x = 1, \\ 0, & \text{если } x \neq 1; \end{cases} \quad \bar{z}(x) = \begin{cases} 1, & \text{если } x \neq 1, \\ 0, & \text{если } x = 1. \end{cases}$$

Функция A будет построена на основании диаграммы переходов $G(P)$ программы P (разд. 2). $G(P)$ состоит из вершин $N(i)$, $1 \leq i \leq s$, и ребер $b(i)$, $0 \leq i \leq s$, $c(i)$, $1 \leq i \leq s$. Ребрам $b(i)$, $c(i)$ соответствуют последовательности Q_i^* , Q_i^{**} Σ -команд и, следовательно, функции $q_r(Q_i^*)$, $q_r(Q_i^{**}) \in Cl(\Sigma^*)^{[r, r]}$. С вершиной $N(i)$ свяжем функции $U^{(i)} \in \mathbf{M}^{[r, r]}$ и $V^{(i)} \in \mathbf{M}^{[r, r]}$, где

$$U^{(0)} = q_r(Q_0^*),$$

$$U_j^{(i)}(X) = z(X_{\alpha(i)}) q_r(Q_i^*)_j(X) + \bar{z}(X_{\alpha(i)}) q_r(Q_i^{**})_j(X),$$

$$1 \leq i \leq s, \quad 1 \leq j \leq r;$$

$$= \begin{cases} q_r(Q_i^*)_j, & \text{если } X_{\alpha(i)} = 1, \\ q_r(Q_i^{**})_j, & \text{если } X_{\alpha(i)} \neq 1, \end{cases} \quad X \in \mathbf{N}^r, \quad 1 \leq i \leq s, \quad 1 \leq j \leq r;$$

$$V^{(0)}(X) \equiv 1,$$

$$V^{(i)}(X) = i^* z(X_{\alpha(i)}) + i^{**} \bar{z}(X_{\alpha(i)}) =$$

$$= \begin{cases} i^*, & \text{если } X_{\alpha(i)} = 1, \\ i^{**}, & \text{если } X_{\alpha(i)} \neq 1, \end{cases} \quad X \in \mathbf{N}^r, \quad 1 \leq i \leq s.$$

Таким образом, если в процессе вычисления при помощи P при достижении вершины $N(i)$ $(R_1, \dots, R_r) = X$, следующей достигнутой вершиной будет $N(V^{(i)}(X))$, при этом $(R_1, \dots, R_r) = U^{(i)}(X)$.

Пусть $R_{r+1}(X, y)$ обозначает индекс i вершины $N(i)$, соответствующей y -й команде выбора, встречающейся при работе P над X , и пусть $R_j(X, y)$, $1 \leq j \leq r$, обозначает содержимое регистров R_j в этот момент — работа P над $X \in \mathbf{N}^n$ начинается с $N(0)$ при $R_i(X, 0) = X_i$, $1 \leq i \leq r$. Тогда содержимое регистров в момент выполнения $(y+1)$ -й команды выбора выражается через содер-

¹⁾ $1(x) = \delta(x, x)$, $z(x) = \delta(x, 1(x))$, $\bar{z}(x) = \delta(0(x), z(x))$. — Прим. ред.

жимое регистров в момент выполнения y -й команды выбора следующим образом:

$$\begin{aligned}
 R_j(X, y+1) &= \sum_{i=0}^s \delta(R_{r+1}(X, y), i) U_j^{(i)}(R_1(X, y), \dots, R_r(X, y)), \\
 &= \begin{cases} q_r(Q_i^*)_j, & \text{если } R_{\alpha(i)}(X, y) = 1 \text{ и } R_{r+1}(X, y) = i, \\ q_r(Q_i^{**})_j, & \text{если } R_{\alpha(i)}(X, y) \neq 1 \text{ и } R_{r+1}(X, y) = i, \end{cases} \\
 &\equiv A_j(R_1(X, y), \dots, R_{r+1}(X, y)), \quad 1 \leq j \leq r, \\
 R_{r+1}(X, y+1) &= \sum_{i=0}^s \delta(R_{r+1}(X, y), i) V^{(i)}(R_1(X, y), \dots, R_r(X, y)), \\
 &= \begin{cases} i^*, & \text{если } R_{\alpha(i)}(X, y) = 1 \text{ и } R_{r+1}(X, y) = i, \\ i^{**}, & \text{если } R_{\alpha(i)}(X, y) \neq 1 \text{ и } R_{r+1}(X, y) = i, \end{cases} \\
 &\equiv A_{r+1}(R_1(X, y), \dots, R_{r+1}(X, y)).
 \end{aligned}$$

Однако так как $q_r(Q_i^*), q_r(Q_i^{**}) \in Cl(\Sigma^*)^{[r, r]}$, то $U^{(i)} \in Cl(\Sigma^*)^{[r, r]}$, $V^{(i)} \in Cl(\Sigma^*)^{[r, 1]}$, а также $A_i \in Cl(\Sigma^*)^{[r+1, 1]}$. Следовательно, $A = (A_1, \dots, A_{r+1}) \in Cl(\Sigma^*)^{[r+1, r+1]}$. Отсюда следует, что

$$(R_1(X, y), \dots, R_{r+1}(X, y)) = A^y I_{r+1}^n(X).$$

Содержимое регистра R_j в момент выполнения m -й команды выбора $I(j_p)$ при работе P над X есть поэтому

$$R_j(X, m) = (A^m I_{r+1}^n(X))_j, \quad 1 \leq j \leq r,$$

и

$$R_{r+1}(X, m) = p = (A^m I_{r+1}^n(X))_{r+1}, \quad \text{ч. т. д.}$$

Отсюда сразу следует

Следствие 1. Пусть B есть (h, Σ) -вычислимая функция в $\mathbf{M}^{[n, m]}$. Тогда

1) существуют числа $\alpha_1, \alpha_2, \dots, \alpha_m$ и функция $A \in Cl(\Sigma^*)^{[r, r]}$, такие, что

$$(B(X))_i \equiv (A^{h(X)} I_r^n(X))_{\alpha_i}, \quad 1 \leq i \leq m;$$

2) существуют функции $R_i(X, y) \in F(\Sigma)$, $1 \leq i \leq m$, такие, что

$$(B(X))_i = R_i(X, h(X)).$$

Из следствия 1 и равенств (2) может быть получена нормальная форма функций из классов $E(\Sigma)_r$. Пусть $f \in E(\Sigma)_r^{[n, 1]}$.

Тогда существуют $r+1$ функций $R_i \in F(\Sigma)^{[n+1, 1]}$, $0 \leq i \leq r$, и число m , так что

$$f(X) = R_r(X, R_{r-1}(X, \dots, R_1(X, R_0(X, m)) \dots)) \quad (6)$$

(фактически мы можем всегда предполагать $m=1$).

Следовательно,

$$E(\Sigma)_\omega \subseteq Cl(F(\Sigma)). \quad (7)$$

Включение (7) будет усилено до равенства в следствии 5 разд. 5.

Из следствия 1.2 и леммы 1 непосредственно вытекает теорема 1.

Из построения A на стр. 104 ясно, что представление функций совместными итерациями, устанавливаемое теоремой 1, зависит от того, содержит ли Σ множество $\{X, +, \delta\}$. Разумеется, всякое исходное множество функций Σ , такое, что $Cl(\Sigma^*) \equiv \{+, X, \delta\}$, дает тот же самый результат. Выбирая Σ более простым множеством функций, можно получить более тонкую классификацию начала цепи $\{E_r\}$, $0 \leq r \leq \omega^2$, однако такой выбор влечет за собой отказ от единой формы получения иерархии с помощью совместной итерации и композиции (теорема 2.1.2).

На основании теоремы 1 мы можем определить без ограничения общности множество Σ -программ как множество $\mathcal{A}(\Sigma) = \bigcup_{r=1}^{\infty} Cl(\Sigma^*)^{[r, r]}$. Если $A \in Cl(\Sigma^*)^{[r, r]}$, то r есть ранг $r(A)$ функции A .

Из равенства (5), следствия 1 и леммы 1 имеем

Следствие 2.

$$E(\Sigma)_0 = Cl(\Sigma^*) \text{ и } F(\Sigma) \subseteq E(\Sigma)_1.$$

5. СВОЙСТВА ПРОГРАММ И КОМПОЗИЦИЯ ФУНКЦИЙ

Для того чтобы построить композицию функций, вычислимых на URM программами, следует иметь в виду, что условие останова определяется однозначно самими программами. Так, чтобы вычислить $f(g(x))$, необходимо отождествить команду останова программы g с начальной командой программы f . Условие останова JM-программы определяется, однако, J -регистром, а не программой. Поэтому возможность композиции функций, вычислимых на JM, зависит от построения программ специального вида, известных как «самоограниченные (SL) программы».

Пусть $A \in M^{[n, m]}$ и $\psi \in Cl(\Sigma^*)^{[p, p]}$, $p > \max(n, m)$. Тогда ψ является SL Σ -программой для A , если существуют числа

$\alpha(1), \dots, \alpha(m)$ и функция γ , определенная на $\mathbf{N}^n$, такие, что

$$\begin{aligned} A_i(X) &\equiv (\psi^{\gamma(X)} I_p^n(X))_{\alpha(i)}, & 1 \leq i \leq m; \\ (\psi^r I_p^n(X))_{\alpha(i)} &\equiv (\psi^{\gamma(X)} I_p^n(X))_{\alpha(i)}, & 1 \leq i \leq m \text{ и } r \geq \gamma(X); \\ \varphi(r, X) &\equiv (\psi^r I_p^n(X))_p = \begin{cases} 0, & \text{если } r \leq \gamma(X), \\ 1, & \text{если } r > \gamma(X). \end{cases} \end{aligned}$$

γ называется **C-пределом** для ψ , а $\varphi(r, X)$ — **сигналом**. Таким образом, ψ обладает «внутренним» пределом для числа итераций, необходимых для того, чтобы завершить вычисление $\psi^{\gamma(X)}$ и, кроме того, получить значение сигнала, изменяющегося от 0 до 1.

SL Σ -программу ψ можно включить в объемлющую программу Γ ; сигнал для ψ служит началом вычисления другой частью программы Γ после того, как вычисление $\psi^{\gamma(X)}$ уже завершено. Так, предположим, что $\Theta \in \mathfrak{A}(\Sigma)^{[r, r]}$. Пусть еще $X \in \mathbf{N}^r$, $Y \in \mathbf{N}^p$ и

$$\begin{aligned} \Gamma_i(X, Y) &\equiv \Theta_i(X) \delta(Y_p, 1) + X_i \delta(Y_p, 1), & 1 \leq i \leq r, \\ \Gamma_{r+i}(X, Y) &\equiv \psi_i(Y), & 1 \leq i \leq p. \end{aligned}$$

Если тогда $Z \in \mathbf{N}^n$, $n < p$, то

$$\Gamma^m(X, I_p^n(Z)) = \begin{cases} (X, (\psi^m I_p^n(Z))_1, \dots, (\psi^m I_p^n(Z))_{p-1}, 0), & \text{если } m \leq \gamma(Z); \\ (X, (\psi^{\gamma(Z)} I_p^n(Z))_1, \dots, (\psi^{\gamma(Z)} I_p^n(Z))_{p-1}, 1), & \text{если } m = \gamma(Z) + 1; \\ (\Theta^{m-\gamma(Z)-1}(X), (\psi^{\gamma(Z)} I_p^n(Z))_1, \dots, (\psi^{\gamma(Z)} I_p^n(Z))_{p-1}, 1), & \text{если } m > \gamma(Z) + 1, \end{cases}$$

т. е. вычисление Θ от X будет эффективным только тогда, когда сигнал для ψ изменяется от 0 до 1.

Заметим, что если A вычисляется на SL Σ -программе с C-пределом g и если существует такая $F \in E(\Sigma)^{[n, 1]}$, что $F \geq g$, то $A(X) \equiv (\psi^{F(X)} I_p^n(X))_{\alpha(i)}$, $1 \leq i \leq m$, а также $A \in E(\Sigma)^{[n, m]}_{r+1}$.

Следующая лемма, устанавливающая свойства SL программ, доказывается стандартным применением вышеизложенных положений.

Лемма 3.

1. Пусть $F_i \in \mathbf{M}^{[n, 1]}$, $1 \leq i \leq m$, вычислимы SL Σ -программами соответственно с C-пределами $f_1, \dots, f_m$, и пусть $\Phi \in Cl(\Sigma^*)^{[m, 1]}$. Тогда $\Phi(F_1, \dots, F_m) \in \mathbf{M}^{[n, 1]}$ и $(F_1, \dots, F_m) \in \mathbf{M}^{[n, m]}$ вычислимы SL Σ -программами с C-пределами $\max(f_1, \dots, f_m) + 1$.

2. Пусть $F = [P, G]$. Если G можно вычислить SL Σ -программой с C -пределом g , то F можно вычислить SL Σ -программой с C -пределом $G + g + 1$.

3. Если $A \in \mathbf{M}^{[m, n]}$, $B \in \mathbf{M}^{[l, m]}$ можно вычислить SL Σ -программами соответственно с C -пределами f, g , то AB можно вычислить SL Σ -программой с C -пределом $fB + g + 1$.

4. Пусть $f \in \mathbf{M}^{[n+2, 1]}$ определена посредством

$$f(X, z, 0) = z, \quad f(X, z, y + 1) = h(X, y, f(X, z, y)).$$

Если h можно вычислить SL Σ -программой с C -пределом g , то f можно вычислить SL Σ -программой с C -пределом

$$K(X, y, z) = \sum_{r=0}^{y-1} g(X, r, f(X, z, r)) + 2y + 1.$$

Индукцией по r из леммы 3.1 мы получаем

Следствие 3. Если $A_1, \dots, A_m \in E(\Sigma)_r^{[n, 1]}$ и $\Phi \in Cl(\Sigma^*)^{[m, 1]}$, то существуют функции $f \in \mathbf{M}^{[n, 1]}$ и $F \in E(\Sigma)_{r-1}^{[n, 1]}$, $f \leq F$, такие, что $(A_1, \dots, A_m)$ и $\Phi(A_1, \dots, A_m)$ вычислимы SL Σ -программами с C -пределами f . Следовательно, $(A_1, \dots, A_m) \in E(\Sigma)_r^{[n, m]}$ и $\Phi(A_1, \dots, A_m) \in E(\Sigma)_r^{[n, 1]}$.

Из леммы 3.2 и следствия 3 индукцией по r получаем, что все функции в $\bigcup_{r < \omega} E(\Sigma)_r$ можно вычислить SL Σ -программами.

Следовательно, индукция по r в лемме 3.3 дает

Следствие 4. Если $A \in E(\Sigma)_r^{[m, n]}$, $B \in E(\Sigma)_s^{[l, m]}$, то $AB \in E(\Sigma)_{r+s}^{[l, n]}$.

Теорема 2.1.1 следует отсюда немедленно.

Из следствия 4 ясно, что класс $E(\Sigma)_\omega$ замкнут относительно композиции.

Следствие 5. $Cl(F(\Sigma)) = E(\Sigma)_\omega$.

Доказательство. Следствие 2 дает $F(\Sigma) \subseteq E(\Sigma)_1 \subseteq \dots \subseteq E(\Sigma)_\omega$. Так как $E(\Sigma)_\omega$ замкнуто относительно композиции, то $Cl(F(\Sigma)) \subseteq E(\Sigma)_\omega$. Однако из (7) разд. 4 $E(\Sigma)_\omega \subseteq Cl(F(\Sigma))$. Следовательно, $Cl(F(\Sigma)) = E(\Sigma)_\omega$, ч. т. д.

Теорема 2.1.2 следует из следствий 2 и 5.

6. НОРМАЛЬНЫЕ МНОЖЕСТВА ФУНКЦИЙ

Пусть S — множество функций одного переменного. Тогда, если f — функция одного переменного, f мажорирует S , если для каждой $g \in S$ существует число n , такое, что $f(x) > g(x)$ для всех $x > n$.

Лемма 4. Предположим, $h \in E(\Sigma)_0^{[1, 1]}$. Пусть $f(0, y) = y + 1$, $f(n+1, y) = h(f(n, y))$, $F_0(n, y) = f(n, y)$, $F_{N+1}(n, y) = f(F_N(n, y), y)$. Если h строго возрастающая и для всякого n и всякой $g \in E(\Sigma)_0^{[n, 1]}$ существует число m , такое, что для всех $X \in \mathbb{N}^n$ справедливо $f(m, \max_i X_i) > g(X)$, то

1) для любых N и n если $k(y) \equiv F_N(n, y)$, то $k \in E(\Sigma)_N^{[1, 1]}$; для каждого n если $l(x, y) \equiv F_N(x, y)$, то $l \in E(\Sigma)_{N+1}^{[2, 1]}$;

2) для всякой $g \in E(\Sigma)_N^{[n, 1]}$ существует число m , такое, что для всех $X \in \mathbb{N}^n$ справедливо $F_N(m, \max_i X_i) > g(X)$;

3) для каждого N если $k(x) \equiv F_N(x, x)$, то $k \in E(\Sigma)_{N+1}^{[1, 1]}$ и k мажорирует $E(\Sigma)_N^{[1, 1]}$.

Доказательство этой леммы легко проводится индукцией по N .

Замечание. 1) Если $X \in \mathbb{N}^n$, то положим $\sum^n X \equiv \sum_{i=1}^n X_i$. Тогда

$$\sum^n \in E(\Sigma)_0^{[n, 1]} \text{ и}$$

$$\sum^n (X) \geq \max(X_i).$$

Следовательно,

$$F_N\left(m, \sum^n X\right) \geq F_N(m, \max X).$$

По следствию 4 $F_N\left(m, \sum^n\right) \in E(\Sigma)_N^{[n, 1]}$.

2) Если Σ — нормальное множество функций, то, так как $Cl(\Sigma^*) = E(\Sigma)_0$, $E(\Sigma)_0^{[1, 1]}$ содержит функцию h со свойствами, сформулированными в лемме 4.

Из леммы 4.3 следует

Следствие 6. Если Σ нормально, то $E(\Sigma)_N \subset E(\Sigma)_{N+1}$, $N = 0, 1, 2, \dots$.

Лемма 5. Если Σ нормально, таковым является и $E(\Sigma)_\omega$.

Доказательство. Так как Σ нормально, существует функция h со свойствами, сформулированными в посылаках леммы 4.

Пусть $h^*(x) \equiv h(x, x)$. Тогда $h^* \in F(\Sigma)^{[1, 1]} \subset E(\Sigma)_\omega$ и является строго возрастающей.

Пусть f^* определяется рекурсией $f^*(0, y) = y + 1$, $f^*(n+1, y) = h^*f^*(n, y)$. Так как по следствию 5 $E(\Sigma)_\omega = Cl(E(\Sigma)_\omega)$, то, для того чтобы доказать нормальность $E(\Sigma)_\omega$, достаточно показать, что для каждой $g \in E(\Sigma)_\omega$ существует число m , такое,

что для всех X справедливо $g(X) < f^*(m, \max X)$. Фактически в силу леммы 4.2 необходимо только показать, что для любых N и n существует m , такое, что $F_N(n, x) < f^*(m, x)$.

Теперь $f(n, x) > n + 1$. Следовательно, $F_0(n, x) \leq F_n(0, x)$. Индукцией по N и m получаем $F_N(n, x) \leq F_{N+n}(0, x)$, $F_m(0, x) < f^*(m, x)$. Следовательно, $F_N(n, x) < f^*(N + n, x)$, ч. т. д.

Теорема 2.2 следует теперь из леммы 5 и следствия 6.

7. ОГРАНИЧЕННАЯ РЕКУРСИЯ

Пусть S — класс функций. Функция f определена ограниченной рекурсией из S , если существуют такие функции $g, h, k \in S$, что

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, y+1) &= h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)), \\ f(x_1, \dots, x_n, y) &\leq k(x_1, \dots, x_n, y). \end{aligned}$$

Из леммы 3 следует, что если $g, h \in E(\Sigma)_\omega$, то можно построить для f SL Σ -программу π . Если Σ нормальное и $k \in E(\Sigma)_\omega$, лемма 4 и следствие 4 позволяют отыскать границу для C -предела π в одном из классов $E(\Sigma)_r$; таким образом, для f имеем:

Лемма 6. Пусть Σ — нормальное множество функций.

1) Если $h \in E(\Sigma)_0^{[n+2, 1]}$, $g \in E(\Sigma)_r^{[n+1, 1]}$ и если $f(X, 0) = g(X)$, $f(X, y+1) = h(X, y, f(X, y))$, то $f \in E(\Sigma)_\mu^{[n+1, 1]}$, $\mu = \max(r, 1)$.

2) Если $h \in E(\Sigma)_s^{[n+2, 1]}$, $s > 0$, $g \in E(\Sigma)_r^{[n, 1]}$ и существует такая функция $k \in E(\Sigma)_t^{[n+1, 1]}$, что для всех $X \in \mathbf{N}^n$ справедливо $f(X, y) \leq k(X, y)$, где $f(X, 0) = g(X)$, $f(X, y+1) = h(X, y, f(X, y))$, то $f \in E(\Sigma)_\mu^{[n+1, 2]}$, $\mu = \max(r, s + t)$.

Пусть $f(X, y) = \sum_{i < y} F(X, i)$, $g(X, y) = \prod_{i < y} F(X, i)$. Из леммы 3.1 следует, что если $F \in E(\Sigma)_s^{[n+1, 1]}$, то $f, g \in E(\Sigma)_s^{[n+1, 1]}$. Таким образом, каждый класс $E(\Sigma)_r$ замкнут относительно операций ограниченного умножения и суммирования. Более того, если Σ — нормальное множество, то класс $E(\Sigma)_\omega$ замкнут относительно ограниченной рекурсии.

Пункты 1, 2, 3 теоремы 2.3 сразу следуют из лемм 5 и 6.

8. ИЕРАРХИЯ ГЖЕГОРЧИКА КЛАССОВ ПРИМИТИВНО РЕКУРСИВНЫХ ФУНКЦИЙ

Для установления результатов этого раздела целесообразно свести совместную итерацию к простой итерации, используя функции, нумерующие пары. Пусть $\bar{\Sigma} \equiv Cl(\Sigma^* \cup \{J, K, L\})$.

Из стандартного представления (равенство (6)) функций $E(\Sigma)_\omega$ вытекает

Лемма 7. Пусть Σ нормально, и пусть $F'(\Sigma)^{[n+1, 1]}$ — множество функций R , определенных на $\mathbf{N}^{n+1}$, где $R(X, y) \equiv K\Phi^y\beta_m I_m^n(X)$, $m \geq n$, $\Phi \in \bar{\Sigma}^{[1, 1]}$. В этом случае $f \in E(\Sigma)_\omega^{[n, 1]}$ тогда и только тогда, когда существуют функции $R_1, \dots, R_s \in F'(\Sigma)$, такие, что

$$f(X) \equiv R_1(X, R_2(X, \dots, R_s(X, 1) \dots)).$$

Отсюда непосредственно следует, что

$$Cl(F'(\Sigma)) = E(\Sigma)_\omega. \quad (8)$$

Далее, так как $K, \beta_m \in E_\omega$, имеем

$$Cl(F'(E_{\omega s})) = E_{\omega(s+1)}. \quad (9)$$

Множество K элементарных по Кальмару функций является наименьшим классом функций, содержащим $x + 1$, x^y и замкнутым относительно подстановки и ограниченной рекурсии (Гжегорчик [1]). Приведенная ниже лемма 8 является аналогом результата Ричи.

Лемма 8. Если $\Sigma = \{+, \times, \delta\}$, то $E_\omega = K$.

Доказательство. (1) $K \supseteq E_\omega$. Мы будем пользоваться леммой 7, чтобы показать, что $F'(\Sigma) \subseteq K$, где $F'(\Sigma)$ есть множество функций $R(X, y) \equiv K\Phi^y\beta_m I_m^n(X)$, $\Phi \in \bar{\Sigma}^{[1, 1]}$.

Во-первых, $J, K, L, +, \times, \delta \in K$. Следовательно, $\bar{\Sigma} \subseteq K$. В таком случае достаточно доказать, что функции T , где

$$T(x, 0) = x, \quad T(x, y + 1) = \Phi T(x, y) \equiv \Phi^{y+1}(x) \quad (10)$$

и $\Phi \in \bar{\Sigma}^{[1, 1]}$, принадлежат $K^{[2, 1]}$. $\Sigma = \{+, \times, \delta\}$ нормально с $h(y) = (y + 1)^2$. Пусть f определяется посредством $f(0, y) = y + 1$, $f(n + 1, y) = h(f(n, y))$. Тогда $f(x, y) \leq 2^{4^x} (y + 1)^{2^{2^x}} \equiv H(x, y)$ и $H \in K^{[2, 1]}$. Так как $J \leq (x + y + 1)^2$, $(x + y + 1)^2 \in Cl(\Sigma^*)^1$, $z \geq Kz$, то для каждой функции $\Phi \in \bar{\Sigma}$ существует такое число k , что $f(k, x) > \Phi(x)$ для всех x . Следовательно,

$$H(ky, x) > f(ky, x) > \Phi^y(x) \equiv T(x, y). \quad (11)$$

Из (10), (11) следует, что T определяется ограниченной рекурсией из элементарных функций и поэтому $T \in K$. Таким образом, $F'(\Sigma) \subseteq K$, и, следовательно, по лемме 7 $K \supseteq E(\Sigma)_\omega$.

(2) $K \subseteq E(\Sigma)_\omega$. Из теоремы 2 получаем, что E_ω замкнут относительно композиции и ограниченной рекурсии. Далее,

¹⁾ В оригинале $J \in Cl(\Sigma^*)$. — Прим. перев.

функции $x + 1$, x^y содержатся в E_0 и E_1 соответственно. Следовательно, $K \subseteq E_\omega$.

На основании (1) и (2) имеем $K = E_\omega$, ч. т. д.

Теперь может быть доказана теорема 3, показывающая, что последовательность $\{E_r\}$, $\omega \leq r \leq \omega^2$, является для $r \geq 3$ уточнением иерархии $\{\mathcal{E}^r\}$ Гжегорчика.

Теорема 3. Если $\Sigma = \{+, \times, \delta\}$, то для $s \geq 1$ $E_{\omega s} = \mathcal{E}^{s+2}$.

Доказательство. Гжегорчик показал, что $\mathcal{E}^3 = K$. Теорема, следовательно, верна для $s = 1$. Мы докажем, что предположение $E_{\omega s} = \mathcal{E}^{2+s}$ влечет $\mathcal{E}^{3+s} \subseteq E_{\omega(s+1)}$ и $\mathcal{E}^{3+s} \supseteq E_{\omega(s+1)}$. Теорема будет отсюда следовать индукцией по s .

(1) Пусть $m = s + 2$, и пусть f_m определена как в (4). Если $g_m(x) = f_m(x + 1, x + 1)$, то по индуктивному предположению $g_m \in E_{\omega s}$.

Пусть $G(y, x) = g_m^y(x)$. Тогда по теореме 2.3.1 $G \in E_{\omega s+1}$. Далее, $f_{m+1}(y, x) \equiv G(2^y, x)$. Однако если $k(y) = 2^y$, то $k \in E_\omega \subseteq E_{\omega s}$. Следовательно, по теореме 2.1.1 $f_{m+1} \in E_{\omega s+1}$. Так как $E_{\omega(s+1)}$ замкнут относительно ограниченной рекурсии и $x + 1$, $U_1^2, U_2^2 \in E_0$, то $\mathcal{E}^{m+1} \subseteq E_{\omega(s+1)}$.

(2) Можно легко показать, что если $h \in \mathcal{E}^n$, то существует x , такое, что для всех y

$$h(y) \leq f_{n+1}(x, y)^1. \quad (12)$$

Предположим, что $E_{\omega s} \subseteq \mathcal{E}^{s+2}$, и пусть $\Phi \in E_{\omega s}^{[1, 1]}$. Тогда из (12) следует, что существует такое число k , что для всех x

$$\Phi(x) \leq f_{m+1}(k, x).$$

По индукции отсюда следует, что

$$\Phi^y(x) \leq f_{m+1}(y + k, x). \quad (13)$$

Однако если $l(y, x) \equiv f_{m+1}(y + k, x)$, то $l \in \mathcal{E}^{m+1}$. А так как $\mathcal{E}^{m+1} \supset \mathcal{E}^m$, то $\Phi \in \mathcal{E}^{m+1}$. (13) определяет $\Phi^y(x)$ ограниченной рекурсией в $\mathcal{E}^{m+1}$. Следовательно, так как $K, \beta_m \in \mathcal{E}^3$, то $F'(E_{\omega s}) \subseteq \mathcal{E}^{m+1}$, а при помощи (9) заключаем, что $E_{\omega(s+1)} = Cl(F'(E_{\omega s})) \subseteq Cl(\mathcal{E}^{m+1}) \subseteq \mathcal{E}^{m+1}$, ч. т. д.

Следствие 7. E_{ω^2} является множеством примитивно рекурсивных функций.

Доказательство. Имеем $P = \bigcup_{n=0}^{\infty} \mathcal{E}^n = \bigcup_{r=0}^{\infty} E_{\omega r}$, ч. т. д.

¹⁾ См., например, [1]. — Прим. перев.

Гжегорчик показал также, что для $n > 2$ класс $\mathcal{E}^{n+1}$ содержит универсальную функцию для класса $\mathcal{E}_1^n$. Обобщая, можно показать, что если Σ — конечное нормальное множество функций, то для $1 \leq r < \omega$ множество $E_{\omega r+1}^{[n+1, 1]}$ содержит универсальную функцию для $E_{\omega r}^{[n, 1]}$.

ЛИТЕРАТУРА

1. Grzegorzczuk A., Some classes of recursive functions, *Rozprawy Matematyczne*, **4** (1953), 1—46. (См. стр. 9—49 настоящего сборника.)
2. Ritchie R., Classes of recursive functions of predictable complexity, Thesis, Princeton.
3. Shepherdson J., Sturgis H., Computability of Recursive Functions, *Journ Assoc. Comp. Mach.*, **10** (1963), 217—255.
4. Robinson R. M., Primitive recursive functions, *Bull. Amer. Math. Soc.*, **53** (1947), 915—942.
5. Péter R., Graphschemata und rekursive Funktionen, *Dialectica*, **12** (1958), 373—393.

ИТЕРАЦИЯ ПРИМИТИВНОЙ РЕКУРСИИ¹⁾

П. Акст

В [2] примитивно рекурсивные функции классифицированы в иерархию $\mathcal{E}^n$, $n = 0, 1, 2, \dots$, основанную на операции ограниченной примитивной рекурсии. В [1] эта иерархия, начинаясь с $\mathcal{E}^4$, конструируется с помощью некоторых методов перечисления, описанных в [4]. В данной работе иерархия примитивно рекурсивных функций строится на еще одной основе, именно на подсчете числа итераций схемы примитивной рекурсии. При этом мы указываем, каким образом она связывается с иерархией из [2] и [1].

Функции f_n и классы $\mathcal{E}^n$ определены в [2]²⁾. (I), ..., (V) — примитивно рекурсивные схемы из [3]. Для $n = 0, 1, 2, \dots$ определим класс K_n примитивно рекурсивных функций следующим образом: $\varphi \in K_n$ тогда и только тогда, когда φ вводится посредством одной из схем (I), (II), (III) или (IV), если данные функции $\psi, \chi_1, \dots, \chi_m$ — уже построенные функции из K_n , или посредством (V), если $n > 0$ и данные функции $\psi, \chi \in K_{n-1}$. Немедленно следует, что $K_n \subseteq K_{n'}$ и что $\bigcup K_n$ является классом примитивно рекурсивных функций.

Теорема 1. Для всех n $f_n \in K_n$.

Доказательство (индукция по n). Рассмотрим сначала $n = 0, 1, 2$.

Если $n = 0$, то $f_n(b, a) = a'$ и $\lambda ba(a') \in K_0$.

Если $n = 1$, то $f_n(b, a) = b + a$. Так как $\lambda ba(b + a)$ задается посредством

$$\begin{cases} b + 0 = b, \\ b + a' = (b + a)' \end{cases}$$

и λbb и $\lambda cc'$ принадлежат K_0 , $\lambda ba(b + a) \in K_1$. Если $n = 2$, то $f_n(b, a) = b' \cdot a'$. Так как $\lambda ba(b' \cdot a')$ задается посредством

$$\begin{cases} b' \cdot 0' = b', \\ b' \cdot a' = b' \cdot a + b' \end{cases}$$

¹⁾ Axt P, Iteration of primitive recursion, *Zeitsch. f. math. Logik und Grundlagen d. Math.*, 11, № 3 (1965), 253—255.

²⁾ Стр. 33 русского перевода. — Прим. ред.

и $\lambda b b'$ и $\lambda b a(b + a)$ принадлежат K_1 , то $\lambda b a(b' \cdot a') \in K_2$. Если $n \geq 2$, допустим (индукционное предположение), что $f_n \in K_n$. Отметим, что $\lambda b 2^b$ задается посредством

$$\begin{cases} 2^0 = 1, \\ 2^{b'} = 2^b + 2^b, \end{cases}$$

так что $\lambda b 2^b \in K_2$. Поэтому если определить

$$\begin{cases} \alpha_{n'}(0, a) = f_n(a', a'), \\ \alpha_{n'}(b', a) = f_n((\alpha_{n'}(b, a))', (\alpha_{n'}(b, a))'), \end{cases}$$

то $f_{n'}(b, a) = \alpha_{n'}(2^b, a)$ и $\alpha_{n'} \in K_{n'}$. Так как $K_2 \subseteq K_{n'}$, $n \geq 2$, мы имеем $f_{n'} \in K_{n'}$. Это завершает доказательство.

Теорема 2. $K_0 \subseteq \mathcal{E}^0$, $K_1 \subseteq \mathcal{E}^1$ и, если $n \geq 2$, $K_n \subseteq \mathcal{E}^{n'}$ и $K_n \not\subseteq \mathcal{E}^n$.

Доказательство. Немедленно следует, что $K_0 \subseteq \mathcal{E}^0$. Чтобы показать, что $K_1 \subseteq \mathcal{E}^1$, достаточно показать, что если φ введена примитивной рекурсией с данными функциями из K_0 , то φ ограничена функцией из $\mathcal{E}^1$. Чтобы сделать это, покажем сначала индукцией по числу применений схем, не выводящих из K_0 , что для любого $\alpha \in K_0$ имеются i , $1 \leq i \leq n$, и k , такие, что $\alpha(a_1, \dots, a_n) \leq a_i^{(k)} = a_i + k$; далее, вводя φ примитивной рекурсией, предположим, что $\psi(a_2, \dots, a_n) \leq a_i^{(k_1)}$, $2 \leq i \leq n$, и $\chi(b_1, \dots, b_{n'}) \leq b_j^{(k_2)}$, $1 \leq j \leq n'$. Индукция по a_1 показывает, что $\varphi(a_1, \dots, a_n) \leq a_1^{(k_2)}$, если $j=1$, $\leq a_i^{(k_1)} + a_i \cdot k_2$, если $j=2$, $\leq a_j^{(k_2)}$, если $2 < j < n$. Таким образом,

$$\varphi(a_1, \dots, a_n) \leq \xi(a_1, \dots, a_n) = a_1^{(k_2)} + a_i^{(k_1)} + a_i \cdot k_2 + a_j^{(k_2)} \text{ и } \xi \in \mathcal{E}^1.$$

Но $K_2 \not\subseteq \mathcal{E}^2$. Из доказательства теоремы 1 следует, что $\lambda a 2^a \in K_2$; в силу [2, теоремы 3.4 и 4.8] для каждой одноместной функции $\varphi \in \mathcal{E}^2$ имеется такое k , что φ ограничена посредством $\lambda a f_3(k, a)$. Далее $f_3(k, a) \leq (a + 2)^r$, где $r = 2^{2^{k+1}}$ [2, стр. 29]¹⁾, и $\lambda a (a + 2)^r$ в свою очередь мажорируется для каждого k функцией $\lambda a 2^a$. Однако для $\varphi \in K_2$ можно показать индукцией по числу применений схем, что имеются значения k и k_j , $j = 1, \dots, n$, и перестановка $a_{i_1}, \dots, a_{i_n}$ чисел $a_1, \dots, a_n$, такие, что для $\rho(a_1, \dots, a_n) = k + a_{i_1} \exp a_{i_1} \dots \exp a_{i_1} \exp a_{i_2} \dots \dots \exp a_{i_n}$ с k_j вхождениями a_{i_j} , $j = 1, \dots, n$, в степень, $\varphi(a_1, \dots, a_n) \leq \rho(a_1, \dots, a_n)$. Функция $\rho \in \mathcal{E}^3$. Следовательно,

¹⁾ См. стр. 34 настоящего сборника. — Прим. ред.

если φ вводится примитивной рекурсией, то функция, ограничивающая φ , существует в $\mathcal{E}^3$. Для случая $n > 2$ может быть использовано [1, Corollary Lemma 3], чтобы показать индукцией по n , что каждая функция из K_n , полученная примитивной рекурсией, ограничивается некоторой функцией из $\mathcal{E}^{n'}$ и, следовательно, что если $\varphi \in K_n$, то $\varphi \in \mathcal{E}^{n'}$. Наконец, мы дополним начатое обсуждение, показав, что для $n \geq 2$ имеется одноместная функция $\gamma_n \in K_n$, которая для каждого k мажорирует $\lambda a f_{n'}(k, a)$ и, следовательно [2, теоремы 3.4 и 4.8], любую одноместную функцию из $\mathcal{E}^n$. Таким образом, для $n \geq 2$ $K_n \not\subset \mathcal{E}^n$. Для $n = 2$ применяется рассуждение, проведенное выше. Для $n \geq 2$ пусть $\gamma_{n'}$ определяется из γ_n посредством $\gamma_2(a) = 2^a$ и

$$\begin{cases} \delta_{n'}(0, a) = \gamma_n(a), \\ \delta_{n'}(b', a) = \gamma_n(\delta_{n'}(b, a)) \end{cases}$$

и $\gamma_{n'}(a) = \delta_{n'}(\kappa(a), a)$, где $\kappa(a) = (a + 4)^r \cdot 2^{a'}$, $r = 2^{2^{a+3}}$. Так как $\kappa \in K_3$, то $\delta_n, \gamma_n \in K_n$ для $n \geq 3$. При этом γ_n — монотонно возрастающая функция. Теперь индукцией по n , $n \geq 3$, показываем, что γ_n ограничивает $\alpha_{n'}$. Если $n = 3$, отметим, что $\alpha_4(a) = f_4(a', a') = \alpha_3(\alpha_3(\alpha_3 \dots \alpha_3(a') \dots))$ с $2^{a'}$ повторениями α_3 (обозначим $\alpha_3^{[2^{a'}]}(a')$), причем $\alpha_3(a') = f_3(a'', a'') \leq (a + 4)^r$, где r определено, как выше. Однако $\lambda a \gamma_3(a)$ ограничивает $2^{a'}$ повторений $(a + 4)^r$ повторений $\lambda a 2^a$. Если $n > 3$, то в силу монотонности γ_n и по индукционному предположению $\alpha_{n'}(a) = \alpha_n^{[2^{a'}]}(a') \leq \gamma_{n-1}^{[2^{a'}]}(a') \leq \gamma_{n-1}^{[\kappa(a)]}(a') = \gamma_n(a)$. Таким образом, для $n \geq 3$ $\gamma_n(a) \geq \alpha_{n'}(a)$, и если $a > k$, то $\gamma_n(a) > f_{n'}(k, a)$.

Следствие. $K_n \subset K_{n'}$, причем $K_n \neq K_{n'}$.

Доказательство. Уже было отмечено, что $K_n \subseteq K_{n'}$. По теореме 2 для всех n $K_n \subseteq \mathcal{E}^{n'}$ и для $n \geq 2$ $K_n \not\subset \mathcal{E}^n$. Так, для $n \geq 1$ $K_{n'} \not\subset K_n$. Так как $\lambda a 2 \cdot a \in K_1$ и так как для каждой одноместной функции $\varphi \in K_0$ имеется такое k , что $\varphi(a) \leq a + k$, $K_1 \not\subset K_0$.

Теорема 3. Для каждого n имеется такое m , что $\mathcal{E}^n \subset K_m$.

Доказательство. В силу [2, теорема 4.12] если $n > 2$, тогда $\mathcal{E}^{n'}$ включает нумерующую функцию для одноместных функций из $\mathcal{E}^n$. Эта нумерующая функция примитивно рекурсивна и, таким образом, для некоторого m принадлежит K_m . Мы можем выбрать $m \geq 2$. Тогда каждая функция из $\mathcal{E}^n$ принадлежит K_m , так как она может быть задана только нумерующей

функцией с помощью функций, нумерующих пары и принадлежащих K_2 . Аналогично [2, теорема 4.12] доказывается, что $\mathcal{E}^4$ включает нумерующие функции для $\mathcal{E}^0$, $\mathcal{E}^1$ и $\mathcal{E}^2$.

ЛИТЕРАТУРА

1. Axt P., Enumeration and the Grzegorczyk hierarchy, *Zeitsch. f. math. Logik und Grundlagen d. Math*, **9**, № 1 (1963), 53—65.
2. Grzegorczyk A., Some classes of recursive functions, *Rozprawy Matematyczne*, **4** (1953), 1—44. (Русский перевод см. стр. 9—49.)
3. Kleene S. C., Introduction to metamathematics, Van Nostrand, 1952. (Русский перевод: Клини С. К., Введение в метаматематику, ИЛ, М, 1957.)
4. Kleene S. C., Extension of an effectively generated class of functions by enumeration, *Colloquium Mathematicum*, **6** (1958), 67—78

ИТЕРАЦИЯ ОТНОСИТЕЛЬНОЙ ПРИМИТИВНОЙ РЕКУРСИИ ¹⁾

П. Акст

В [1] примитивно рекурсивные функции классифицированы в иерархию типа ω в соответствии с числом применений схемы примитивной рекурсии в примитивно рекурсивном определении функции. Мы распространяем этот результат, давая релятивизированную версию этой иерархии. Метод, используемый нами, дает другое доказательство теоремы иерархии из [1]. Аналоги дополнительной информации, предоставляемые методом из [1], будут приведены ниже.

Приспособив обозначение и определение из [1] для $l, n \geq 0$ и произвольной l -местной функции κ , определим K_n^κ как класс функций, примитивно рекурсивных относительно κ , которые вводятся одной из примитивно рекурсивных схем $(), (I), \dots, (IV), (V)$ из [2]²⁾, где в схеме $() \Theta = \{\kappa\}$, в схеме (IV) вспомогательные функции $\psi, \chi_1, \dots, \chi_m$ — уже построенные элементы из K_n^κ и в схеме (V) $n > 0$ и вспомогательные функции $\psi, \chi \in K_{n-1}^\kappa$.

Очевидно, что $\bigcup_n K_n^\kappa$ есть множество всех функций, примитивно рекурсивных относительно κ , и что $K_n^\kappa \subseteq K_{n+1}^\kappa$. Чтобы установить иерархию, мы покажем с помощью двух теорем, что $K_n^\kappa \neq K_{n+1}^\kappa$.

¹⁾ Axt P., Iteration of relative primitive recursion, *Math. Annalen*, 167, № 1 (1966), 53—55.

²⁾ Приведем схемы из [2], определяющие функции, примитивно рекурсивные относительно совокупности $\Theta = \{\Theta_1(a_1, \dots, a_{m_1}), \dots, \Theta_l(a_1, \dots, a_{m_l})\}$:

$$() \quad \varphi(a_1, \dots, a_{m_l}) = \Theta_i(a_1, \dots, a_{m_l});$$

$$(I) \quad \varphi(a) = a';$$

$$(II) \quad \varphi(a_1, \dots, a_n) = q;$$

$$(III) \quad \varphi(a_1, \dots, a_n) = a_i;$$

$$(IV) \quad \varphi(a_1, \dots, a_n) = \psi(\chi_1(a_1, \dots, a_n), \dots, \chi_m(a_1, \dots, a_n));$$

$$(V) \quad \begin{cases} \varphi(0, a_2, \dots, a_n) = \psi(a_2, \dots, a_n), \\ \varphi(a'_1, a_2, \dots, a_n) = \chi(a_1, \varphi(a_1, a_2, \dots, a_n), a_2, \dots, a_n). \end{cases}$$

Теорема 1. Если для некоторого n $K_n^x = K_{n+1}^x$, то любая функция, примитивно рекурсивная относительно x , принадлежит K_n^x .

Доказательство. Мы покажем индукцией по r , что для любого r из $\varphi \in K_{n+r}^x$ следует $\varphi \in K_n^x$; для каждого r проводится индукция по числу итераций примитивно рекурсивных схем (), (I), ..., (V) в определении φ в K_{n+r}^x (глубина φ в K_{n+r}^x) для случаев, соответствующих схеме, посредством которой φ была введена. Случаи 0, ..., 4 очевидны. Для случая 5 заметим, что вспомогательные функции примитивной рекурсии принадлежат K_{n+r}^x , следовательно, и K_n^x по допущению индукции по r . Таким образом, $\varphi \in K_{n+1}^x$, и следовательно, по условию теоремы $\varphi \in K_n^x$. Итак, для всех r $K_n^x = K_{n+r}^x$, т. е. каждая функция, примитивно рекурсивная относительно x , принадлежит K_n^x .

Теорема 2. Каково бы ни было n , существует функция, примитивно рекурсивная относительно x , которая не принадлежит K_n^x .

Доказательство. Производящая функция $\bar{x}$ [3, стр. 680] является возрастающей по всем своим переменным и примитивно рекурсивной относительно x . Например, для $l = 2$

$$\bar{x}(b, a) = \prod_{i < b} p_i \exp \prod_{j < a} p_j \exp (x(i, j) + 1).$$

Для удобства нашего доказательства при $l = 0$ положим, что $\bar{x}$ есть $\lambda a \prod_{i < a} p_i$. Для данной l -местной функции x не будет никакой путаницы, если мы обозначим $\bar{x}(a, \dots, a)$ через $\bar{x}(a)$. Теперь определим возрастающие функции γ_0^* и γ_0 , которые примитивно рекурсивны относительно x ,

$$\begin{cases} \gamma_0^*(0, a) = \bar{x}(a'), \\ \gamma_0^*(b', a) = \bar{x}(\gamma_0^*(b, a)), \\ \gamma_0(a) = \gamma_0^*(a, a). \end{cases}$$

Доказательство проводим индукцией по n . Для $n = 0$ мы покажем индукцией по глубине $\varphi \in K_0^x$, что существует число b , такое, что если $a = \max[a_1, \dots, a_k]$, то $\varphi(a_1, \dots, a_k) < \gamma_0^*(b, a)$. Среди случаев, соответствующих схемам (), (I), ..., (IV), посредством которых φ была введена в K_0^x , случаи 0, ..., 3

очевидны. В случае 4 $\varphi(a_1, \dots, a_k) = \psi(\chi_1(a_1, \dots, a_k), \dots, \chi_m(a_1, \dots, a_k))$. По допущению индукции по глубине φ и вследствие монотонности γ_0^* существуют числа $b_0, b_1, \dots, b_m$, такие, что для $a \geq \max[a_1, \dots, a_k]$, $i = 1, \dots, m$,

$$\psi(a_1, \dots, a_k) < \gamma_0^*(b_0, a),$$

$$\chi_i(a_1, \dots, a_k) \leq \gamma_0^*(b_i, a).$$

Заметим, что $b = \max[b_0, b_1, \dots, b_m]$ может быть использовано для замещения в этих неравенствах b_0 и b_i . Таким образом, $\psi(c_1, \dots, c_m) < \gamma_0^*(b, \gamma_0^*(b, a))$ для всех $c_i \leq \gamma_0^*(b, a)$, $i = 1, \dots, m$. Отсюда

$$\begin{aligned} \psi(\chi_1(a_1, \dots, a_k), \dots, \chi_m(a_1, \dots, a_k)) < \\ < \gamma_0^*(b, \gamma_0^*(b, a)) \leq \gamma_0^*(2b + 2, a), \end{aligned}$$

если $a = \max[a_1, \dots, a_k]$ ¹⁾. Таким образом, для любой $\varphi \in K_0^\kappa$ и для $a = \max[a_1, \dots, a_k] > b$ имеем $\varphi(a_1, \dots, a_k) < \gamma_0(a)$. Итак, $\gamma_0 \notin K_0^\kappa$, γ_0 — возрастающая функция, примитивно рекурсивная относительно κ . Следовательно, базис индукции по n доказан.

Для $n \geq 0$ определим функции γ_{n+1}^* и γ_{n+1} следующим образом:

$$\begin{cases} \gamma_{n+1}^*(0, a) = \gamma_n(a), \\ \gamma_{n+1}^*(b', a) = \gamma_n(\gamma_{n+1}^*(b, a)), \\ \gamma_{n+1}(a) = \gamma_{n+1}^*(a, a) + 1. \end{cases}$$

Пусть H_{n+1}^κ — класс функций, полученных примитивной рекурсией с вспомогательными функциями из K_n^κ . Предположим по индукции, что γ_n^* и γ_n возрастающие и примитивно рекурсивные и что для каждой $\varphi \in K_n^\kappa$ существует число b , такое, что если $a = \max[a_1, \dots, a_k]$, то $\varphi(a_1, \dots, a_k) < \gamma_n^*(b, a)$, и если к тому же a является достаточно большим, то $\varphi(a_1, \dots, a_k) < \gamma_n(a)$.

¹⁾ В оригинале $\gamma_0^*(b, \gamma_0^*(b, a)) = \gamma_0^*(2b + 1, a)$, что, как легко видеть, неверно:

$$\gamma_0^*(b, \gamma_0^*(b, a)) = \underbrace{\bar{\kappa}(\bar{\kappa}(\dots \bar{\kappa}(\bar{\kappa}(\bar{\kappa}(\dots \bar{\kappa}(a') \dots)))}_{b+1 \text{ раз}} + 1) \dots).$$

В то же время

$$\gamma_0^*(2b + 1, a) = \underbrace{\bar{\kappa}(\bar{\kappa}(\dots \bar{\kappa}(a') \dots))}_{2b+2 \text{ раза}}.$$

В индукционном шаге рассмотрим два случая в зависимости от того (для $\varphi \in K_{n+1}^*$), φ принадлежит H_{n+1}^* или нет.

Случай 1. $\varphi \in H_{n+1}^*$. По предположению индукции для вспомогательных функций ψ, χ примитивной рекурсии, из которых φ была получена, существуют константы b_0 и b_1 , такие, что

$$\psi(a_2, \dots, a_k) < \gamma_n^*(b_0, a) \quad \text{и} \quad \chi(a_1, c, a_2, \dots, a_k) < \gamma_n^*(b_1, a),$$

если $a = \max[a_1, c, a_2, \dots, a_k]$. Далее, используя монотонность γ_n^* , для $i = 1, \dots, k$ и для любого b получим $a_i < \gamma_n^*(b, a)$; таким образом, $\varphi(1, a_2, \dots, a_k) = \chi(0, \psi(a_2, \dots, a_k), a_2, \dots, a_k) < < \gamma_n^*(b_0 + b_1 + 1, a)$ для $a = \max[a_2, \dots, a_k]$. Индукцией по a_1 , положив $d = b_0 + b_1 + 1$ ¹⁾, получим

$$\begin{aligned} \varphi(a'_1, a_2, \dots, a_k) &= \chi(a_1, \varphi(a_1, a_2, \dots, a_k), a_2, \dots, a_k) < \\ &< \gamma_n^*(da'_1, a) < \gamma_n^*(da', a), \end{aligned}$$

если $a = \max[a_1, a_2, \dots, a_k]$. Теперь $\gamma_n^*(da', a) < \gamma_n^*(da', da') = \gamma_n^*(da')$ и, как легко можно проверить, γ_{n+1}^* является возрастающей, так как γ_n является также возрастающей, а $da' < \gamma_{n+1}^*(d, a)$. Итак, $\gamma_n^*(da') < \gamma_n^*(\gamma_{n+1}^*(d, a)) = \gamma_{n+1}^*(d', a)$. Таким образом, для любой $\varphi \in H_{n+1}^*$ существует число b , такое, что $\varphi(a_1, \dots, a_k) < \gamma_{n+1}^*(b, a)$, если $a = \max[a_1, \dots, a_k]$.

Случай 2. $\varphi \notin H_{n+1}^*$. В этом случае φ вводится посредством одной из схем (), (I), ..., (IV) с вспомогательными функциями, принадлежащими H_{n+1}^* . Доказательство, подобное тому, которое было проведено в шаге индукции в базисе с γ_{n+1}^* , играющей роль γ_0^* , показывает, что для каждой функции φ , явно определяемой через функции из H_{n+1}^* , существует число b , такое, что

$$\varphi(a_1, \dots, a_k) < \gamma_{n+1}^*(b, a), \quad \text{если} \quad a = \max[a_1, \dots, a_k].$$

Учитывая случаи 1 и 2 и то обстоятельство, что класс функций, явно определяемых через функции из H_{n+1}^* , совпадает с K_{n+1}^* , получаем, что $\gamma_{n+1}^* \notin K_{n+1}^*$. Для проведения индукции

¹⁾ В оригинале в качестве d выбрано $\max[b_0, b_1] + 1$. Однако в этом случае не проходит базис индукции ($a'_1 = 1$) ввиду

$$\gamma_n^*(da'_1, a) = \gamma_n^*(d, a) = \gamma_n^*(1 + \max[b_0, b_1], a) < \gamma_n^*(b_0 + b_1 + 1, a).$$

нужно учесть, что γ_{n+1} — функция, примитивно рекурсивная относительно κ , а так как γ_{n+1}^* возрастающая, то γ_{n+1} является возрастающей. Доказательство на этом заканчивается.

Следствие. Для всех $n \geq 0$ $K_n^\kappa \subset K_{n+1}^\kappa$.

Доказательство. По теореме 2 не существует n , такого, что каждая функция, примитивно рекурсивная относительно κ , принадлежит K_n^κ . Отсюда по теореме 1 ни для какого n $K_n^\kappa = K_{n+1}^\kappa$ не имеет места.

ЛИТЕРАТУРА

1. Axt P., Iteration of primitive recursion, *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 11, № 3 (1965), 253—255. (Русский перевод см. в настоящем сборнике, стр. 114—117.)
2. Kleene S. C., Extension of an effectively generated class of functions by enumeration, *Coll. Math.*, 6 (1958), 67—78.
3. Kleene S. C., Recursive functions and intuitionistic mathematics, *Proc. International Congress of Mathematicians*, 1 (1950), 679—685.

О ДВУХ ПОДХОДАХ К КЛАССИФИКАЦИИ РЕКУРСИВНЫХ ФУНКЦИЙ

А. А. Мучник

Один способ классификации рекурсивных функций восходит к истокам теории алгоритмов и связан с порождением тех или иных классов рекурсивных функций из множеств (обычно конечных) достаточно простых исходных функций посредством различных вычислимых операций. Так строятся классификации Петер, Гжегорчика, Акста, Клини. При этом обычно возникает растущая иерархия (цепочка) классов по некоторому трансфинитному типу. Такой подход можно было бы назвать аналитическим. (В последнее время с помощью алгебраических методов изучаются классы рекурсивных функций, образующие подалгебры в алгебрах относительно тех или иных вычислимых операций. При этом обычно идут «сверху — вниз — в стороны» в отличие от аналитического подхода «снизу — вверх».) В сравнительно недавнее время появились работы, в которых вычислимые функции классифицируются по сложности их вычисления, а также работы, устанавливающие связь между аналитическим и «сложностным» определением классов рекурсивных функций (см. статью Р. В. Ричи в настоящем сборнике). В нашем добавлении мы рассматриваем этот вопрос в общей постановке.

Одна из проблем, поставленных А. Гжегорчиком в статье «Некоторые классы рекурсивных функций» (см. настоящий сборник, стр. 9—49), состоит в выяснении того, можно ли в определении классов $\mathcal{E}^n$ ($n = 0, 1, 2, \dots$) исключить операцию ограниченной примитивной рекурсии. Эта проблема была решена положительно для классов $\mathcal{E}^n$ при $n \geq 3$ Рёдингом [7], а для $\mathcal{E}^2$ — Марченковым [4]. (Вопрос о классах $\mathcal{E}^0$ и $\mathcal{E}^1$ до сих пор открыт). Доказательства у этих авторов используют сложную технику. Интерпретация классов $\mathcal{E}^n$ ($n \geq 2$) как классов функций, вычислимых на машинах Тьюринга при некоторых ограничениях на длину используемой ленты, позволяет дать простое доказательство этих результатов. Кроме того, мы опишем общий метод для нахождения конечного базиса (относительно суперпозиции, а также других операций) во многих классах рекурсивных функций.

Будем говорить, что функция $\varphi(n, x_1, \dots, x_k, y)$ является квазиуниверсальной (к. у.) для k -местных функций алгебры $\mathfrak{A} = \langle G, *, O_1, \dots, O_l, \dots \rangle$ (здесь G есть некоторый класс функций; $*$ — операция суперпозиции; $O_1, \dots, O_l, \dots$ — некоторые вычислимые операции) относительно системы функций H , если для всякой функции ¹⁾ $\lambda x_1 \dots x_k f(x_1, \dots, x_k) \in G$ существует число n_0 и функция $\lambda x_1 \dots x_k g(x_1, \dots, x_k) \in Cl_\sigma(H)$ (здесь $Cl_\sigma(H)$ обозначает множество функций, порождаемых с помощью операций сигнатуры σ из H), такие, что

$$f(x_1, \dots, x_k) \equiv \varphi(n_0, x_1, \dots, x_k, g(x_1, \dots, x_k)).$$

Если, кроме того, $f(x_1, \dots, x_k) \equiv \varphi(n_0, x_1, \dots, x_k, h(x_1, \dots, x_k))$ для всякой функции $h(x_1, \dots, x_k) \geq g(x_1, \dots, x_k)$, то функцию φ назовем максимально универсальной (м. у.) для G относительно H . Класс вычислимых функций Φ назовем σ -Q-U (σ -M-U-) полным, если 1) он содержит к. у. (м. у.) функцию $\varphi(n, x, y)$ для класса Φ_1 всех одноместных функций от x из Φ относительно конечного множества $H \subset \Phi$; 2) класс Φ замкнут относительно операций из σ ²⁾; 3) класс Φ содержит константу 0, $x + 1$ и какую-нибудь тройку функций $I(x, y)$, $K(z)$, $L(z)$, нумерующих пары, т. е. таких, что ³⁾

$$I(Kz, Lz) = z; \quad KI(x, y) = x; \quad LI(x, y) = y.$$

Очевидно, что всякий σ -M-U-полный класс является σ -Q-U-полным.

Теорема. *Всякий σ -Q-U-полный класс содержит конечный базис относительно σ .*

Доказательство. Выберем в качестве базиса B в G множество функций $H \cup \{\varphi, 0, x + 1, I\}$. Пусть функция $f(x_1, \dots, x_k) \in G$. Обозначим через $I_m(x_1, \dots, x_m)$ функцию

$$x = I(x_1, I(x_2, \dots, I(x_{m-1}, x_m) \dots)).$$

Тогда $x_1 = Kx = K_1x$, $x_2 = KLx = K_2x$, $x_3 = KLLx = K_3x, \dots$, $x_{m-1} = KL^{m-2}x = K_{m-1}x$, $x_m = L^{m-1}(x) = K_mx$, где $L^i(x)$ обозначает i -ю итерацию функции $L(x)$. Рассмотрим функцию $f_1(x) = f(K_1x, \dots, K_mx)$. Очевидно, согласно определению σ -Q-U-полного класса, $f_1 \in G$. Но тогда найдутся число n_0 и функция

¹⁾ Выражение $\lambda x_1 \dots x_k f(x_1, \dots, x_k)$ (λ -конверсия), введенное Чёрчем, обозначает функцию f в отличие от значения $f(x_1, \dots, x_k)$, которое она принимает на наборе аргументов $(x_1, \dots, x_k)$. Иногда там, где это не будет вызывать недоразумений, мы для краткости в обозначении функций будем опускать приставку $\lambda x_1 \dots x_k$.

²⁾ Операции из σ не обязательно определены для всех наборов функций из Φ (например, примитивная рекурсия).

³⁾ Некоторые скобки в указанных здесь и далее равенствах опущены, но могут быть очевидным образом восстановлены.

$g(x)$, являющаяся суперпозицией функций из H , такие, что $f_1(x) = \varphi(n_0, x, g(x))$, т. е. $f_1 \in Cl_\sigma(B)$. Легко видеть, что

$$f(x_1, \dots, x_m) = f_1[I_m(x_1, \dots, x_m)],$$

а так как I_m принадлежит $Cl_\sigma(B)$, то теорема доказана. Из этой теоремы можно извлечь разнообразные следствия. Сначала докажем, что каждый класс $\mathcal{E}^n$ при $n \geq 2$ является $*$ - M - U -полным. Условия 2 и 3, как известно, для этих классов выполняются.

Рассмотрим некоторую универсальную машину Тьюринга (МТ) $\mathfrak{M}$ с алфавитом $\{0, 1, \rho, \lambda\}$ (λ — пустой символ, ρ — разделительный символ), которая по входному слову $P\rho X$ вычисляет функцию $\varphi_P(x)$, где P — код программы некоторой машины Тьюринга, а X — двоичный код числа x (оба кода в алфавите $\{0, 1\}$). Пусть n — номер P_n при лексикографической нумерации всех слов в $\{0, 1\}$. Будем считать n также номером машины с кодом программы P_n . Положим $\varphi(n, x) = \varphi_{P_n}(x)$; очевидно, что $\varphi(n, x)$ — частично рекурсивная функция (ч.р.ф.), универсальная для одноместных ч.р.ф. $\varphi(x)$. Введем функцию $\varphi^*(n, x, y)$, равную результату вычисления на $\mathfrak{M}$ с исходной записью $P_n\rho X$, если вычисление укладывается на ленте длины $[\log_2 y]$, в противном случае $\varphi^*(n, x, y)$ положим равной 0. Очевидно, что можно построить МТ, вычисляющую $\varphi^*(n, x, y)$ и использующую при этом не более чем $[\log_2 n] + [\log_2 x] + [\log_2 y] + 3$ квадрата ленты. Согласно результатам Ричи (см. его статью в настоящем сборнике), функция $\lambda x y \varphi^*(n, x, y) \in F' = \mathcal{E}^2$. Нетрудно показать, что она является м.у. функцией для одноместных функций алгебры $\langle \mathcal{E}^2, * \rangle$ относительно $H = \{(x+1)^2\}$. Действительно, согласно Ричи, $\lambda x_1 \dots x_k f(x_1, \dots, x_k) \in F'$ тогда и только тогда, когда существует МТ с двоичным рабочим алфавитом (не считая ρ и λ), вычисляющая $f(x_1, \dots, x_k)$ на ленте длины, не превосходящей $C_1 \left(\sum_{i=1}^k [\log_2 (x_i + 1)] \right) + C_2$, где C_1 и C_2 — постоянные,

зависящие от f . Следовательно, $f \in F' = \mathcal{E}^2$ тогда и только тогда, когда существуют такие натуральные числа n_0, m, c_1 и c_2 , что $f(x) = \varphi(n_0, x, g(x))$, где $g(x) \geq c_1 x^m + c_2$. Но очевидно, что всякая функция вида $c_1 x^m + c_2$ мажорируется некоторой итерацией полинома $(x+1)^2$, откуда следует, что $\varphi^*(n, x, y)$ — м.у. функция для $\langle \mathcal{E}^2, * \rangle$ относительно $\{(x+1)^2\}$. Вместо $(x+1)^2$ можно взять $f_2(x, y)$ — функцию Гжегорчика. Аналогично устанавливается, что $\varphi^*(n, x, y)$ является м.у. функцией для алгебры $\langle \mathcal{E}^r, * \rangle$ ($r \geq 2$) относительно $\{f_r(x, y)\}$, где $f_r(x, y)$ есть r -я функция Гжегорчика (стр. 33). Вместо $f_r(x, y)$ можно взять r -ю

функцию Аккермана, определенную следующей схемой:

$$\begin{aligned} p_1(x) &= x + 2; \quad p_2(x) = 2x; \\ \begin{cases} p_{r+1}(0) = 1, \\ p_{r+1}(x+1) = p_r(p_{r+1}(x)) \end{cases} & \text{ для } r \geq 2. \end{aligned}$$

Известно, что $\lambda x p_r(x) \in \mathcal{E}^r$ ($r = 0, 1, 2, \dots$) и что при определении классов $\mathcal{E}^r$ можно исходить из функций Аккермана (см. [15]).

Строгое доказательство этого факта проводится индукцией по r : положим $g_r(x) = f_r(x, x)$. Нужно установить,

1) что каждая функция f из $\mathcal{E}_1^r$ мажорируется¹⁾ некоторой функцией $c_1 \cdot g_{r,s}(x) + c_2$ (или $c_1 \cdot p_{r,s}(x) + c_2$), где $g_{r,1}(x) = g_r(x)$, а $g_{r,s+1}(x) = g_r(g_{r,s}(x))$ ($p_{r,s}$ определяются аналогично);

2) что каждая функция $f(x)$ из $\mathcal{E}_1^r$ вычислима на некоторой машине Тьюринга с лентой длины, не превосходящей $c_1 \cdot g_{r,s}(x) + c_2$, где s, c_1 и c_2 зависят от f и $\mathfrak{M}$;

3) что при $r \geq 2$ для всех s, c_1 и $c_2 \geq 0$ справедливо утверждение

$$\lambda n x \varphi^*(n, x, c_1 \cdot g_{r,s}(x) + c_2) \in \mathcal{E}^r.$$

Эти факты полностью доказаны Я. Б. Казановичем [1] (их можно установить методом Ричи; см. его статью в настоящем сборнике).

Следствие (Марченков, Рёдинг). *В каждом классе $\mathcal{E}^r$ ($r \geq 2$) существует конечный базис относительно суперпозиции.*

Напрашивается следующая классификация примитивно рекурсивных функций по типу ω^2 , близкая к предложенной Казановичем [1].

Определим класс $F_{r,s}$ функций так: $\lambda x_1 \dots x_k f(x_1, \dots, x_k) \in F_{r,s}$ тогда и только тогда, когда существуют натуральные числа n_0, s, c_1 и c_2 , такие, что

$$f(x_1, \dots, x_k) = \varphi^*(n_0, I_k(x_1, \dots, x_k), c_1 \cdot g_{r,s}(I_k(x_1, \dots, x_k)) + c_2),$$

т. е. функция f вычислима на некоторой машине Тьюринга с двоичным кодированием чисел, но произвольным рабочим алфавитом и с рабочей зоной (т. е. длиной ленты), не большей чем²⁾

$$\log(g_{r,s} I_k(x_1, \dots, x_k) + c);$$

¹⁾ Напомним, что $\mathcal{E}_1^r$ есть класс всех одноместных функций из $\mathcal{E}^r$.

²⁾ От функций $I(x, y)$ ($I_n(x_1, \dots, x_n)$) здесь достаточно потребовать взаимно однозначного вычислимого отображения множества пар (n -ок) натуральных чисел в N , что легко можно осуществить конкатенацией с разделительным символом двоичных кодов чисел x и y ($x_1, \dots, x_n$). Очевидно, что определенные так функции $I(x, y)$ и $I_n(x_1, \dots, x_n)$ принадлежат классу $F' = \mathcal{E}^2$.

при $r \geq 2$ $\bigcup_{s=1}^{\infty} F_{r,s} = \mathcal{E}^r$. При $r = 2$ получаем систему классов $\{F_{2,s}\}$, образующую иерархию подклассов $\mathcal{E}^2$, близкую к той, которая определяется в [2] посредством вычисления функций из $\mathcal{E}^2$ на многоголовочных автоматах. Некоторое представление о классе $F_{2,s}$ дает верхняя (достижимая) оценка скорости роста: каждая функция $f(x_1, \dots, x_n)$ из $F_{2,s}$ ограничена некоторым полиномом от $x_1, \dots, x_n$ степени, не большей чем $2n \cdot s$. При $r = 3$ $F_{r,s} = F_s$, где F_s — класс Ричи ($s = 1, 2, \dots$). На первый взгляд кажется, что такая классификация зависит от выбора функций $g_{r,s}(x)$, однако иерархия $\{F_{r,s}\}$ возникает естественным, «предсказуемым» (predictable) образом, подобно тому, как у Акста [6] (см. перевод в настоящем сборнике) строится система классов K_n ($n = 0, 1, 2, \dots$). Исходными являются чрезвычайно простые функции:

$$0, \quad x + 1, \quad U_1(x, y) = x, \quad U_2(x, y) = y.$$

Они образуют систему B_0 . ($K_0 = Cl_*(B_0)$.) Применим ко всем парам функций $f, g \in K_0$ от n и $n + 2$ переменных соответственно операцию примитивной рекурсии. Обозначим полученное множество через B_1 . Замкнем его по суперпозиции. Полученный класс обозначим через K_1 . Затем опять применим один раз примитивную рекурсию к парам (f, g) из K_1 и т. д. Акст доказал, что $\bigcup_n K_n$ равно Π_1 — классу всех примитивно рекурсивных функций [6].

Парсонс [24] явно выписал базис (из сравнительно простых функций) для $\mathcal{E}^3$, а Швихтенберг [8] доказал, что $K_n = \mathcal{E}^{n+1}$ для всех $n \geq 3$. (Включение $\mathcal{E}^n \subseteq K_n \subseteq \mathcal{E}^{n+1}$ доказал Акст.) В. А. Козмидиadi [3] построил иерархию растущих автоматов, вычисляющих примитивно рекурсивные функции, индуцирующие иерархию Гжегорчика — Акста, начиная с $\mathcal{E}^3$. Система классов $F_{r,s}$ является подразбиением системы $\{K_r\}$ ($r \geq 3$) или $\{\mathcal{E}^r\}$ ($r \geq 2$)¹⁾. Положим

$$B'_0 = B_0 \cup \{\varphi^*(n, x, y), I, K, L\}.$$

(При этом выберем I, K и L достаточно «простыми» из $\mathcal{E}^2$; см. например, статью Гжегорчика в настоящем сборнике, стр. 10.) $K'_0 = Cl_*(B'_0) = \mathcal{E}^2$. Исходя из B'_0 , будем строить классы $K_{r,s}$, где s является «глубиной суперпозиций» функций из B'_r , а $K'_r = Cl_*(B'_r)$. Здесь надо оговорить, что в аргументы допускается

¹⁾ По аналогии с тем, что классы $K_r = \mathcal{E}^{r+1}$ ($r \geq 3$) замкнуты относительно ограниченной рекурсии, можно было бы ожидать, что классы $F_{r,s}$ замкнуты относительно ограниченной суперпозиции. Однако Казанович заметил, что это не так. Существует предикат $\lambda nxy T(n, x, y) \in F_{3,1}$, такой, что $\lambda nx T(n, x, p_{3,s}(x)) \in F_{3,s} \setminus F_{3,1}$ ($s > 1$), хотя $p_{3,s} \in F_{3,s}$ и $T(n, x, y) \leq 1$.

подстановка функций $U_1(x, y)$, $U_2(x, y)$ и $I_m(x_1, \dots, x_m)$, которая не учитывается при подсчете глубины суперпозиции функций из K'_r , а также фиксировать некоторый способ перехода от K'_r к B'_{r+1} путем построения функции $\tau_r(x) = \tau(r, x)$, универсальной для одноместных функций из K'_r . ($B'_{r+1} = B'_0 \cup \{\tau(r, x)\}$.) Тот факт, что для всякой конечной системы функций $A = \{\varphi_1, \dots, \varphi_l\}$ с помощью одной примитивной рекурсии и нескольких суперпозиций относительно $A \cup B_0 \cup \{x+y, x-y, x \cdot y\}$ можно построить функцию, универсальную для всех одноместных функций из $Cl_*(A)$, устанавливается с помощью техники, изложенной в работах [10] или [11, § 5.2] (можно даже вместо $Cl_*(A)$ взять замыкание A по операциям суперпозиции и ограниченной примитивной рекурсии). Можно показать, что класс $K_{r,s}$ совпадает с классом $F_{r+2, s+c}$, где c — постоянная, зависящая от способа построения универсальной функции для K'_r ($r, s = 0, 1, 2, \dots$). Таким образом, класс $F_{r+2, s}$ является классом функций, полученных из B'_0 с помощью r примитивных рекурсий и $s + \text{const}$ «существенных» суперпозиций (т. е. без учета подстановок функций I_m, U_1, U_2) типа $p_r(p_r(\dots p_r(x) \dots))$.

Итак, общий способ построения иерархий рекурсивных функций предсказуемым образом таков: некоторые «достаточно простые» функции выбираются в качестве исходных — это система B . Наиболее простой вычислимой операцией, порождающей новые функции из B , является суперпозиция. $Cl_*(B)$ разбивается на подклассы по числу использованных суперпозиций. Затем для класса $Cl_*(B)$ строится универсальная функция с помощью примитивной рекурсии. Смысл примитивной рекурсии как операции над f и g состоит в том, что при этом один из аргументов «регулирует» число суперпозиций (некоторого специального вида) функции g . (Это обстоятельство особенно наглядно проявляется в частном случае примитивной рекурсии — итерации.) Все другие виды рекурсий, сводящиеся к примитивно рекурсивным, связаны с теми или иными суперпозициями функций f и g в схеме рекурсии (см. [11, § 3; 13, § 3—8]¹⁾). Дальнейшее раз-

¹⁾ Часто применяются некоторые специальные виды примитивной рекурсии: ограниченные μ , Σ и Π , а также их обобщения, описанные в статье Гжегорчика из этого сборника. Однако и в этих схемах один из аргументов опять-таки регулирует суперпозиции, но только одних и тех же функций (для каждой операции — своя функция): $\mu(x, y)$, $x + y$, $x \cdot y$, x^y и т. д. Операции ограниченного μ и навешивания ограниченных кванторов легко сводятся к ограниченному суммированию и суперпозиции при наличии некоторых простых функций.

Широко известно, что оператор обращения μ вместе с суперпозицией и примитивной рекурсией порождает класс всех ч. р. ф., но по схеме, задающей ч. р. ф. f , невозможно эффективно распознать, является ли f общерекурсивной функцией [13, § 22].

витие классификации рекурсивных функций связано с операцией двойной рекурсии R_2 , при которой один из аргументов «регулирует» число применений операции примитивной рекурсии над преобразуемыми функциями. Известно, что для класса одноместных функций, примитивно рекурсивных относительно конечного множества функций $A = \{\varphi_1, \dots, \varphi_l\}$, можно построить универсальную функцию, являющуюся дважды рекурсивной относительно A (см. [10; 11, § 5]). (Таким путем Клини, используя систему O' обозначений для конструктивных ординалов, основанную на примитивной рекурсии, пытался построить растущую, т. е. вполне упорядоченную, иерархию всех общерекурсивных функций [10]. Однако на этом пути встретились серьезные трудности [12].) Далее идут операции тройной, ... k -кратной, ... и трансфинитных рекурсий [13, § 10—15]. Возможность эффективного продолжения классификации связана с построением достаточно далеких эффективных унивалентных систем обозначений для конструктивных ординалов [10]. Проблема оказалась тесно связанной с теорией доказательств, а именно с иерархией арифметических теорий [16, 17]. Действительно, всякая частично рекурсивная функция может быть задана системой равенств термов с одним выделенным термом, соответствующим вычисляемой функции¹⁾ $\varphi(\vec{x})$. В нормальной форме Клини $\varphi(\vec{x}) = \rho[\mu y T(\vec{x}, y)]$, где ρ и T примитивно рекурсивны. Общерекурсивность $\varphi(\vec{x})$ означает, что $\forall \vec{x} \exists y T(\vec{x}, y)$. Если эта формула выводима в классической арифметике, то $\varphi(\vec{x})$ называется доказуемо рекурсивной. Теорию таких функций развил Фишер [18]. Однако ими никак не исчерпывается множество всех общерекурсивных функций, как показывает обычная диагональная процедура, поскольку множество выводимых в классической арифметике формул вида $\forall \vec{x} \exists y T(\vec{x}, y)$ с примитивно рекурсивным T является рекурсивно перечислимым. Истинность формулы $\forall \vec{x} \exists y T(\vec{x}, y)$ можно пытаться установить посредством вывода в теории, являющейся расширением классической арифметики посредством однократного, а затем двух, трех и т. д. применений конструктивного правила Карнапа и трансфинитных пополнений арифметики. При этом возникает картина, совершенно аналогичная расширяющейся иерархии классов общерекурсивных функций Клини. Подробное исследование трансфинитных расширений классической и конструктивной арифметических теорий проведено А. Г. Драгалиным [17].

¹⁾ Через $\vec{x}$ обозначен набор аргументов $(x_1, \dots, x_n)$, где $n = 1, 2, \dots$

Представляют интерес некоторые проблемы, связанные с двойной рекурсией (проблемы 6 и 9 статьи Гжегорчика, стр. 46). Операция ограниченной («bounded») двойной рекурсии R_2-b над функциями G , H , h и E определяется так:

$$F(0, x) = h(x),$$

$$F(n+1, 0) = H(n+1, F(n, F(n, 0)), F(n, 0)),$$

$$F(n+1, x+1) = G(n+1, x, F(n+1, x), F(n, F(n, x+1))),$$

$$F(n, x) \leq E(n, x).$$

(Без последнего неравенства имеем обычную упрощенную схему двойной рекурсии R_2 .)

Нетрудно показать, что операция R_2-b , будучи применена к функциям G , H , h и E из класса $\mathcal{E}^r$, дает функцию из класса $\mathcal{E}^{r+1}$. Марченков установил, что замыкание класса $\mathcal{E}^r$ относительно операции R_2-b (класс $\tilde{\mathcal{E}}^{r+1}$) есть в точности подкласс класса $\mathcal{E}^{r+1}$, состоящий из функций, ограниченных функциями из $\mathcal{E}^r$ (так, например, $\tilde{\mathcal{E}}^3$ есть класс функций из $\mathcal{E}^3$, ограниченных функциями из $\mathcal{E}^2$ или, проще говоря, полиномами).

Верны строгие включения

$$\mathcal{E}^r \subset \tilde{\mathcal{E}}^{r+1} \subset \mathcal{E}^{r+1}.$$

Таким образом, класс Π_1 примитивно рекурсивных функций замкнут относительно ограниченной двойной рекурсии. Интересно отметить, что классы $\tilde{\mathcal{E}}^r$ ($r \geq 3$) не имеют конечного базиса относительно суперпозиций. (Этот факт установлен студентом МГУ В. Альперовичем в 1968—1969 гг.) Аналогичные результаты должны иметь место для ограниченных k -кратных рекурсий.

Классификацию Гжегорчика можно продолжить, взяв последовательность функций $\{h_{n(u)}(x)\}$ ($u = 0, 1, 2, \dots$) из [10], где $n(0) = 1$, $n(u+1) = 2^{n(u)}$, а через C_u обозначается класс функций, примитивно рекурсивных относительно $h_u(x)$ ($n(u)$ является обозначением для ординала, соответствующего натуральному числу u в системах O и O' Клини [10]). Однако можно построить последовательность значительно более простых, чем $h_{n(u)}(x)$, функций $e_u(x)$, порождающих классы $C_{n(u)}$ посредством лишь суперпозиций и примитивных рекурсий, основываясь на скорости роста $e_u(x)$, причем $e_{u+1}(x)$ будет получаться из $e_u(x)$ с помощью одной двойной рекурсии. Идея построения $e_u(x)$ такова: $e_1(x) = B(x, x)$, где $B(n, x)$ — функция Аккермана, а n — число примитивных рекурсий, необходимых для получения функции $\lambda x B(n, x)$. Далее берем итерации функций $e_1(x)$: $f(0, x) = e_1(x)$, $f_1(n+1, x) = e_1[f(n, x)]$. Пусть $f_1(x) = f(x, x)$ (диагональная про-

цедура). Далее полагаем

$$f_2(0, x) = f_1(x), \quad f_2(n+1, x) = f_1(f_2(n, x)), \quad f_2(x) = f_2(x, x)$$

(опять «диагональ»). Подобным же образом определяются функции $f_k(x)$. Полагаем $e_2(x) = f_x(x)$ («диагональ» на следующей ступени); функции $e_u(x)$ для $u \geq 3$ строятся аналогично. Этот метод получения все более быстро растущих функций дает «мажорантную» иерархию Фефермана [12].

Классы $C_{n(u)}$ разбивают класс Π_2 по количеству использованных двойных рекурсий. Эти построения можно, конечно, продолжить и дальше по конечным и бесконечным ординалам [13, § 11—13]. Идея Клини, вероятно, состояла в том, что всякой общерекурсивной функции объективно присуща некоторая «сложность», которую можно оценить конструктивным ордина-

лом (а не его обозначением). Акст в [9] доказал¹⁾, что $\bigcup_{u=1}^{\infty} C_{n(u)} = \Pi_2$ (Π_2 — класс дважды рекурсивных функций, который, как легко установить, не содержит конечного базиса относительно примитивно рекурсивных операций).

Другой возможный путь продолжения классификации Π_1 аналогичен методу Акста определения K_n с заменой примитивной рекурсии на двойную. Получающиеся при этом классы совпадают с классами $C_{n(u)}$, а эти последние замкнуты относительно ограниченной двойной рекурсии. Отсюда следует, что в Π_2 нет конечного базиса относительно операций $*$ и R_1 (примитивной рекурсии), т. е. проблема 9 в статье Гжегорчика (стр. 46) решается отрицательно. Строгое доказательство этих фактов связано с весьма громоздкой техникой. (Здесь опять-таки можно использовать м. у. для $\langle C_{n(u)}, *, R_1 \rangle$ относительно $H = \{e_u(x)\}$ функцию $\varphi^*(n, x, z)$.) Продолжение классификации связано с k -кратными рекурсиями.

Хотя выше было отмечено, чем интересны классы кратно рекурсивных функций, однако практически математики и вычислители, как правило, имеют дело с функциями, не выходящими

¹⁾ В [9] получены следующие результаты: 1) для всякого ординала $\alpha < \omega^2$ и любых обозначений y и z в O' для α $C_y = C_z$, 2) $\bigcup_{|y| < \omega} C_y = \Pi_2$ и

3) $\bigcup_{|y| < \omega^2} C_y = \Pi_3$ (Π_3 — класс всех 3-рекурсивных функций), где $|y|$ есть ординал, обозначенный в O' числом y . Акст же установил, что для некоторых ординалов $\alpha \leq \omega^3$ классы C_y могут отличаться друг от друга для разных y , таких, что $|y| = \alpha$. Этот результат был значительно усилен Феферманом [12], доказавшим, что $\bigcup_{|y| = \omega^3} C_y$ является классом всех общерекурсивных функций.

за пределы Π_1 и даже $\mathcal{E}^3$. Это дает нам основания вернуться к рассмотрению проблем, связанных с этими классами.

Иной, чем описанный выше подход к классификации Π_1 по типу ω^2 , предложен в статье Клива из данного сборника. Его классы функций $E_{\omega r+s+1}$ ($r, s = 0, 1, 2, \dots$) определяются предсказуемым образом посредством ограничений на число операций SS -машины Шепердсона — Стургиса [14], налагаемых функциями из класса $E_{\omega r+s}$. Для $s = 0$ число операций ограничивается постоянной, зависящей от вычисляемой функции. Когда определены все классы $E_{\omega r+s}$ при данном r , Клив полагает

$E_{\omega(r+1)} = \bigcup_{s=1}^{\infty} E_{\omega r+s}$ и рассматривает SS -машины, элементарными операциями которых являются функции из $E_{\omega(r+1)}$. В качестве исходных операций для SS -машин классов $E(\Sigma)_r$ берется множество $Cl_*(\Sigma^*)$, где $\Sigma^* = \Sigma \cup \{U_i(x_1, \dots, x_n) = x_i, 1 \leq i \leq n\}$. Если $\Sigma = \{+, \times, \delta\}$, где

$$\delta(x, y) = \begin{cases} 1, & \text{если } x = y, \\ 0, & \text{если } x \neq y, \end{cases}$$

то $E_{\omega r} = \mathcal{E}^{r+2}$, где $r \geq 1$ (основной результат). Открытым остается вопрос о соотношении между классами $E_{\omega r+s}$ и $F_{r,s}$, определенными в нашей статье.

Далее мы рассмотрим классы сравнительно просто вычисляемых рекурсивных функций. Пусть Q — класс функций $f(x_1, \dots, x_n)$, вычисляемых на подходящих машинах Тьюринга с длиной ленты, не большей, чем некоторый полином $P_f(\log(x_1 + 1), \dots, \log(x_n + 1))$ (P_f зависит от f).

Теорема. *Класс Q является M - U -полным¹⁾ и замкнут относительно ограниченной рекурсии.*

Набросок доказательства. Очевидны включения $\mathcal{E}^2 \subseteq Q \subseteq F_1$. Поэтому условие 3 M - U -полноты выполнено. Далее, пусть $f(y)$ и $g(x)$ — две одноместные функции из Q . Докажем, что $f(g(x))$ также принадлежит Q (в случае многоместных функций доказательство аналогично). Вычисление $f(g(x))$ происходит так: сначала МТ вычисляет $g(x)$ на ленте длины $\leq P_g(x)$. Поскольку результат вычисления укладывается на этой ленте, то

$$g(x) \leq 2^{cP_g(\log_2(x+1))} \leq 2^{c_1 \log_2^r(x+1) + c_2},$$

¹⁾ Здесь и далее подразумевается, что $\langle Q, * \rangle$ является M - U -полной алгеброй, т. е. под M - U -полнотой класса функций подразумевается $*$ - M - U -полнота.

где c, c_1, r и c_2 зависят только от g и МТ. Затем МТ вычисляет функцию f от $g(x)$, используя при этом ленту длины

$$l \leq 2^{c_3 \log_2^t(g(x)+1)+c_4} \leq 2^{c_5 \log_2^{rt}(x+1)+c_6},$$

где постоянные c_3, c_4, t, r, c_5 и c_6 зависят от f, g и МТ. Итак, $\lambda x f(g(x)) \in Q$. Введенная ранее функция $\varphi^*(n, x, y)$ служит м. у. функцией для Q относительно $\{x(x)\}$, где $x(x) = (x+2)^{\log_2(x+2)}$, так как итерациями $x(x)$ можно промажорировать любую экспоненту от полинома от $\log_2(x+1)$, а, с другой стороны, $\lambda x x(x) \in Q$. Итак, Q есть $M-U$ -полный класс. Пусть $f(x, y)$ и $h(x)$ — функции из Q и

$$g(0) = c_0; \quad g(x+1) = f(x, g(x)); \quad g(x) \leq h(x).$$

Для вычисления $g(0)$, т. е. по существу для записи c_0 , требуется зона МТ порядка $\log c_0$ (c_0 может быть произвольно большим числом). Для вычисления $g(1)$ МТ вычисляет $f(0, g(0))$, для чего ей нужна рабочая зона, не большая, чем $P_f(0, \log_2(g(0)+1))$. (Полином P_f можно выбрать монотонным с целыми положительными коэффициентами.) Перед вычислением $g(x+1)$ на ленте МТ записаны двоичные коды x и $g(x)$, причем длина кода $g(x)$ не превосходит $\log_2(h(x)+1)+1$, так как $g(x) \leq h(x)$. Вычисление $f(x, g(x))$ поэтому потребует ленты длиной не более чем $P_f(\log_2(x+1), \log_2(h(x)+1)) + \log_2(x+1) + \log_2(h(x)+1) + c \leq P(\log_2(x+1))$, так как $h(x)$, как и всякая функция из Q , мажорируется некоторой функцией вида $2^{\tilde{P}_h(\log_2(x+1))}$, где $\tilde{P}_h$ — полином, зависящий от h .

Теорема доказана.

Теорема. Класс ТР функций, вычислимых на машине Тьюринга с полиномиальным замедлением, т. е. за время, не превосходящее некоторый полином (зависящий от выбранной функции) от аргументов функции, при унарном кодировании значений вычисляемых функций, является $M-U$ -полным. Класс ТР замкнут относительно ограниченной примитивной рекурсии.

Доказательство. Известно, что $ТР \equiv \mathcal{E}^2$. (Открытым остается вопрос: равны ли эти классы?) Поэтому в ТР содержатся функции I, K, L . Легко проверить замкнутость ТР относительно суперпозиций. Далее пусть $\lambda x h(x)$ и $\lambda x y f(x, y)$ принадлежат ТР и

$$g(0) = c_0, \quad g(x+1) = f(x, g(x)), \quad g(x) \leq h(x).$$

Вычисление $g(x+1)$ производится на МТ, которая предварительно вычислила $g(0), g(1), \dots, g(x)$, и к этому моменту на ленте МТ записаны коды x и $g(x)$. Машина начинает вычис-

лать $f(x, g(x))$, для чего ей требуется время $t_{x+1} \leq P_f(x, g(x)) \leq P_f(x, h(x))$, где P_f — монотонный полином, но так как $h(x) \leq P_1(x)$, то $t_{x+1} \leq P_2(x)$. Время вычисления $g(x+1)$, таким образом, не превосходит $\sum_{z \leq x} P_2(z) = P_3(x)$ (P_1 , P_2 и P_3 — некоторые полиномы).

Рассмотрим функцию

$$\varphi^*(n, x, t) = \begin{cases} \varphi_n(x), & \text{если для вычисления } \varphi_n(x) \text{ на } n\text{-й МТ} \\ & \text{достаточно времени } t; \\ 0 & \text{в противном случае.} \end{cases}$$

Подходящая МТ с тремя головками производит вычисление $\varphi^*(n, x, t)$ за время $T \leq ct \cdot \log_2 n$. Поэтому $\varphi^* \in \text{ТР}$ (при переходе к универсальной МТ с одной головкой время увеличивается не более чем в квадрат [19]). Нетрудно показать, что φ^* является м.у. функцией для ТР относительно $H = \{(x+1)^2\}$. Теорема доказана.

Теорема. Класс ТРЛ функций $\{f\}$, вычисляемых на машинах Тьюринга с k -ичным ($k \geq 2$) кодированием чисел за время, не превосходящее некоторый полином (зависящий от f) от длин кодов аргументов, является М-У-полным.

Этот класс «изоморфен» классу словарных (в алфавите из k букв) примитивно рекурсивных функций, вычисляемых с полиномиальным замедлением. Класс ТРЛ на самом деле не зависит от выбора k при $k \geq 2$. М.у. функцией для него служит $\varphi^*(n, x, t)$ относительно $H = \{(x+1)^{\log_2(x+1)}\}$. Доказательство аналогично предыдущему.

Теорема. Класс ЛР функций $f(x_1, \dots, x_n)$, вычисляемых на многоленточных МТ с k -ичным кодированием ($k \geq 2$) чисел за время $T(x_1, \dots, x_n) \leq c \cdot \max_i \{x_i \log_2^r(x_i + 1)\} + c_1$ и таких, что $f(x_1, \dots, x_n) \leq c_2 \cdot \max_i \{x_i \log_2^s(x_i + 1)\} + c_3$, где c, c_1, r, c_2, c_3 и s — постоянные, зависящие от f и МТ, является М-У-полным.

Из результата Хенни и Стирнза (см. их статью в настоящем сборнике) следует, что для каждой такой МТ существует двуленточная МТ с алфавитом $\{0, 1, \rho, \lambda\}$, которая моделирует работу первой МТ за время $CT \log T$. Поэтому ясно, что в определении класса ЛР вместо многоленточных многоголовковых машин можно взять двуленточные МТ с одной головкой на каждой ленте. Теперь построим трехленточную МТ $\mathfrak{M}_3$, универсальную для двуленточных, записывая в двоичном коде программы двуленточных машин на третьей ленте. Номер n двуленточной

машины обычным образом определяется двоичной записью ее программы. Тогда $\mathfrak{M}_3$ моделирует работу двуленточной машины с номером n с замедлением не более чем в $c \log_2 n$ раз. Рассмотрим функцию $\varphi^*(n, x, t)$, модифицированную так, что n теперь является номером двуленточной МТ. Эта функция вычислима на пятиленточной МТ $\mathfrak{M}$ за время $T \leq ct \log_2 n$ ¹⁾, т. е. она принадлежит LP. Можно показать, что φ^* является м. у. функцией для LP относительно $H = \{x \log_2(x + 2) + 1\}$, а также замкнутость LP относительно суперпозиции и наличие в LP функций I, K, L , т. е. M - U -полноту LP.

Каждый из приведенных выше классов можно разбить на подклассы по числу «существенных» суперпозиций, подобно тому, как классы K'_r и $\mathcal{E}'_r$ подразбиваются на подклассы $K_{r,s}$.

Вышеуказанные примеры укладываются в некоторую общую схему. Рассмотрим какую-нибудь функцию $\lambda \vec{x} Sl^k(n, \vec{x})$ сложности вычисления k -местной частично рекурсивной функции (ч. р. ф.) $\lambda \vec{x} \varphi(n, \vec{x})$, удовлетворяющую аксиомам Блюма (см. статью М. Блюма в настоящем сборнике, стр. 401—422), где $\varphi(n, \vec{x}) = \varphi_n(\vec{x})$ — универсальная ч. р. ф., а $\vec{x} = (x_1, \dots, x_k)$ ($k = 1, 2, 3, \dots$). Введем предикаты $T(n, \vec{x}, y)$ и $S(n, \vec{x}, y)$. $T(n, \vec{x}, y)$ означает, что $Sl(n, \vec{x}) = y$, а $S(n, \vec{x}, y) \equiv Sl(n, \vec{x}) \leq y$ (если $\varphi(n, \vec{x})$ не определена, то условно полагаем $Sl(n, \vec{x}) = \infty$). Пусть $\bar{\varphi}(n, \vec{x}, y)$ — ч. р. ф., такая, что $T(n, \vec{x}, y) \rightarrow \bar{\varphi}(n, \vec{x}, y) = \varphi(n, \vec{x})$ и $\bar{\varphi}(n, \vec{x}, y) = 0$, если $\neg T(n, \vec{x}, y)$, а

$$\varphi^*(n, \vec{x}, y) = \begin{cases} \varphi(n, \vec{x}), & \text{если } S(n, \vec{x}, y), \\ 0, & \text{если } \neg S(n, \vec{x}, y). \end{cases}$$

Из аксиом Блюма следует, что $\bar{\varphi}$ и φ^* общерекурсивны.

Пусть Γ — некоторый класс ч. р. ф. Класс Γ и функций $\lambda \vec{x} Sl^k(n, \vec{x})$ ($k = 1, 2, 3, \dots$) определяют два класса ч. р. ф.

$$K_1(\Gamma, Sl^k) \text{ и } K_2(\Gamma, Sl^k).$$

¹⁾ МТ $\mathfrak{M}$ должна иметь две ленты, моделирующие две ленты двуленточной МТ $\mathfrak{M}_2$: одну ленту, где записана программа $\mathfrak{M}_2$ (номер ее n), и одну ленту, на которой записан двоичный код числа t . С помощью еще одной ленты можно производить в двоичном коде подсчет на последних двух лентах машины $\mathfrak{M}$ числа k операций машины $\mathfrak{M}_2$ за $c_1 k$ операций машины $\mathfrak{M}$, связанных с прибавлением 1 к двоичному коду k , каждое из которых производится после моделирования на $\mathfrak{M}$ очередной операции машины $\mathfrak{M}_2$. Поэтому машина $\mathfrak{M}$ имеет пять лент.

K_1 есть класс ч. р. функций

$$\{\lambda \vec{x} f(\vec{x}) \mid (\exists n_0, \exists \gamma \in \Gamma) (f(\vec{x}) = \varphi(n_0, \vec{x}) \& T(n_0, \vec{x}, \gamma(\vec{x})))\},$$

а

$$K_2 = \{\lambda \vec{x} f(\vec{x}) \mid (\exists n_0, \exists \gamma \in \Gamma) (f(\vec{x}) = \varphi(n_0, \vec{x}) \& S(n_0, \vec{x}, \gamma(\vec{x})))\}.$$

Класс Γ назовем классом точных функций сложности для K_1 и классом верхних границ сложности для K_2 . Если для функции $\bar{\varphi}_k(n, \vec{x}, y)$ ($\varphi_k^*(n, \vec{x}, y)$) найдется вычисление, такое, что его функция сложности (некоторая верхняя граница функции сложности) принадлежит Γ , то функцию $\bar{\varphi}_k$ (φ_k^*) мы назовем универсальной (максимально универсальной) для k -местных функций класса K_1 (K_2) относительно Γ .

Если к тому же класс $\Gamma \subset K_1$ ($\Gamma \subset K_2$) и порождается системой функций $H \subseteq \Gamma$ посредством операций $*$, $O_1, \dots, O_l, \dots$, то $\varphi(\varphi^*)$ назовем к. у. (м. у.) функцией для K_1 (K_2) относительно H и сигнатуры $[*, O_1, \dots, O_l, \dots]$. Если класс Γ таков, что K_1 (K_2) содержит систему B_1 функций 0 , $x + 1$, $\bar{\varphi}_1$ (φ_1^*) и I , K , L (т. е. эти функции могут быть вычислены с допустимой сложностью) и замкнут относительно суперпозиций, то K_1 (K_2) также порождается операциями $*$, $O_1, \dots, O_l, \dots$ из $H \cup \{B_1 \setminus \{K, L\}\}$. Исследование классов точных функций сложности (конструируемых в терминологии Стирнза, Харманиса и Льюиса) для различных вычислений представляет большой интерес, хотя их полное описание обычно дать не удастся (по этому поводу см. [21—23]). Значительно легче описываются классы верхних границ (мажорант) для допустимых сложностей вычисления. (Эти классы не обязательно состоят из *всех* мажорант для функций сложности) Например, для класса $\mathcal{E}^3$ элементарных функций таким является

класс $\left\{ 2^{2^{\dots^{2^{\max(x_1, \dots, x_k)}}}} \right\}$ относительно времени или памяти (длины ленты) вычисления, а в качестве H можно выбрать $\{2^*\}$. Обычно класс Γ состоит из монотонных функций и сравнительно просто определяется аналитически, а класс K_2 характеризуется скоростью роста функций из Γ . Это обстоятельство, быть может, впервые проявилось в работе Аккермана (1927 г.), где была построена вычислимая функция, мажорирующая любую примитивно рекурсивную функцию и носящая имя автора. С тех пор на протяжении 40 лет основанные на скорости роста соображения неоднократно встречались во многих работах (например, в работах Гжегорчика (1953 г.), Б. А. Трахтенброта [5] (1956 г.), Ричи (1962—1963 гг.), В. А. Козмидиади и С. С. Марченкова (1965—1967 гг.), Я. Б. Казановича (1967 г.)), связанных с классификацией рекурсивных функций и вычислений.

Таким образом, класс функций, вычислимых не более чем с заданной сложностью, характеризуется быстротой роста функций сложности.

Отметим, что наибольший интерес представляют такие классы функций сложности вычислений, которые инвариантны относительно выбора возможно более широкого класса вычислительных устройств (или вычислительных алгоритмов). Такими, например, являются классы, определяемые порядком используемой при вычислении памяти, выраженной в битах (например, длина ленты машины Тьюринга), или классы TP , TPL и т. п. Основная трудность применения описанной выше схемы состоит в таком определении сложности вычисления, при котором все функции класса Γ порождаются конечной системой функций H , $K_2 \equiv \Gamma$ и в классе K_2 содержится м. у. относительно H функция. До сих пор не удалось применить эту схему к классам $\mathcal{E}^0$ и $\mathcal{E}^1$ Гжегорчика, классу функций, элементарных по Сколему, классу функций, вычислимых с линейным замедлением на многоленточных машинах Тьюринга, и некоторым другим. Вероятно, здесь требуются более тонкие методы.

ЛИТЕРАТУРА

1. Казанович Я. Б., Классификация примитивно рекурсивных функций с помощью машин Тьюринга, Проблемы кибернетики, вып. 22, М., 1970.
2. Козмидиачи В. А., Марченков С. С., О многоголовочных автоматах, Проблемы кибернетики, вып. 21, М., 1969, 127—158.
3. Козмидиачи В. А., О бесконечных автоматах, Проблемы кибернетики, вып. 23 (в печати).
4. Марченков С. С., Устранение схем рекурсий в классе $\mathcal{E}^2$ Гжегорчика, Матем. заметки, 5, № 5 (1969), 561—568.
5. Трахтенброт Б. А., Сигнализирующие функции и табличные операторы, Уч. зап. Пенз. пед. ин-та, IV (1956), 75—87.
6. Axt P., Iteration of primitive recursion (См. стр. 114—117 настоящего сборника.)
7. Rödding D., Über die Eliminierbarkeit von der Definitionsschemata in der Theorie der rekursiven Funktionen, Zt. math. Log. Grundlag. Math., 10, № 4 (1964), 315—330.
8. Schwichtenberg H., Recursionstiefe und Grzegorzcyk-Hierarchy, Arch. math. Log. Grundlagenforsch., B. 12, H. 1—2, 85 (1969).
9. Axt P., Note on the 3-recursive functions, Zt. math. Log. Grundlag. Math., 7, № 1 (1961), 97—98.
10. Kleene S. C., Extension of an effectively generated class of functions by enumeration, Colloquium mathem., 6 (1958), 67—78.
11. Мальцев А. И., Алгоритмы и рекурсивные функции, «Наука», М., 1965, стр. 105—110.
12. Feferman S., Classification of recursive functions by means of hierarchies, Trans. Amer. Math. Soc., 104 (1962), 101—122.
13. Петер Р., Рекурсивные функции, ИЛ, М., 1954.
14. Shepherdson J. C., Sturgis H. E., Computability of recursive functions, Jour. ACM, 10 (1963), 217—255.
15. Ritchie R. W., Classes of recursive functions based on Ackermann's function, Pacif. J. Math., 15, № 3 (1965), 1027—1044.

16. Feferman S., Transfinite recursive progressions of axiomatic theories, *Jour. Symb. Log.*, 27, № 3 (1962), 259—316.
17. Драгалин А. Г., Трансфинитные пополнения арифметики, *ДАН СССР*, 189 (1969), № 3.
18. Fisher P., Theory of provable recursive functions, *Trans. Amer. Math Soc.*, 117 (1965), 494—520.
19. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, 117 (1965), 285—306. (Русский перевод: Хартманис Д., Стернс Р., О вычислительной сложности алгоритмов, Кибернетический сборник (новая серия), вып. 4, 1967.)
20. Aho A., Ullman J., The theory of languages, *Math. Syst. Theory*, 2, № 2 (1967), 97—125. (Русский перевод: Ахоу А., Ульман Д., Теория языков. Обзор, Кибернетический сборник (новая серия), вып. 6, 1969.)
21. Фрейвальд Р. В., О порядке роста точных временных сигнализирующих для тьюринговых вычислений, *Алгебра и логика, Семинар*, 5 (1966), № 5, 85—94.
22. Трахтенброт Б. А., О нормированных сигнализирующих для тьюринговых вычислений, *Алгебра и логика, Семинар*, 5 (1966), № 6, 61—70.
23. Stearns R. E., Hartmanis J., Lewis P. M., Hierarchies of memory limited computations, IEEE Conf. Rec. on Switching Circuit Theory and Logical Design, Ann Arbor, Mich., Oct. 1965. (См. стр. 301—319 настоящего сборника.)
24. Parsons Ch., Hierarchies of primitive recursive functions, *Zeitsch. f. math. Logik und Grundlagen d. Math.*, 14, № 4 (1968), 357—376.

II. О СЛОЖНОСТИ ВЫЧИСЛЕНИЙ НА МАШИНАХ ТЬЮРИНГА

ВЫЧИСЛЕНИЯ В РЕАЛЬНОЕ ВРЕМЯ И РЕКУРСИВНЫЕ ФУНКЦИИ, НЕ ВЫЧИСЛИМЫЕ В РЕАЛЬНОЕ ВРЕМЯ ¹⁾

Х. Ямада

1. ВВЕДЕНИЕ

В результате последних достижений в технике цифровых вычислительных машин эти машины становятся все более и более сложными по конструкции. Становится невозможным изучить все логические свойства вычислительной машины. Отсюда как следствие все возрастающее внимание к теории автоматов — абстрактных моделей вычислительных машин. Поскольку теория автоматов — абстрактная теория, она может и не давать результатов, непосредственно переносимых на машины, но должна давать углубленное понимание природы рассматриваемых проблем. Хороший обзор теории автоматов можно найти в [1].

Общеизвестно, что можно рассматривать машины Тьюринга как математические модели цифровых вычислительных машин [2]. Дело в том, что мы всегда можем, по крайней мере теоретически, построить эквивалентную машину Тьюринга для любой цифровой вычислительной машины [3]. Однако именно в силу своей большой общности машины Тьюринга не являются наилучшей формой для изучения всех аспектов цифровых вычислительных машин. Например, когда мы рассматриваем специальную вычислительную проблему, вообще говоря, трудно, а порой невозможно дать априорную оценку числа шагов, необходимого для получения решения. Из этих соображений многие исследователи работали на разных машинах Тьюринга, каждая из которых логически определяется в терминах некоторых ограничений [4—7]. Эти ограничения делают машины подкласса намного слабее, чем наиболее общие машины Тьюринга. С другой стороны, в силу того, что эти ограничения дают возможность использовать некоторые дополнительные свойства машин Тьюринга, общие для всего подкласса, мы можем исследовать эти машины более основательно. Более того, подкласс машин Тьюринга часто

¹⁾ Y a m a d a H., Real-time computation and recursive functions not real-time computable, *IRE Trans. El. Comp.*, EC-11 (1962), 753—760.

дает нам модель цифровой вычислительной машины, более реалистичную в частных аспектах.

Настоящая работа, которая является попыткой построить математическую модель для цифровых машин, касается класса многоленточных машин Тьюринга, которые с некоторыми ограничениями применяются для вычислений некоторого класса функций в реальное время. Употребляя цифровую вычислительную машину, важно знать время, необходимое для вычисления данной функции. Однако, как известно автору, не было попыток исследовать общую теорию такого рода¹⁾. В качестве начала в этом направлении автор исследовал вычисления в реальное время некоторых рекурсивных функций посредством подкласса машин Тьюринга. Эта работа не претендует на полноту, и необходимы дальнейшие исследования. Тем не менее результаты предыдущих исследований положены в основу этой статьи и сведены в разд. 2. В качестве примера в разд. 3 рассмотрено построение некоторой машины из рассматриваемого класса для вычисления полиномов. Для детального рассмотрения интересующегося читателя отсылаем к первоисточнику [8]. В разд. 4 этой статьи мы даем дальнейшую разработку теории и устанавливаем основную теорему, согласно которой вне зависимости от правил вычисления существует подкласс рекурсивных функций, которые не могут быть вычислены в реальное время. Автор полагает, что читатель знаком с основами теории машин Тьюринга и рекурсивных функций [2, 3].

2. ВЫЧИСЛЕНИЯ В РЕАЛЬНОЕ ВРЕМЯ С ПОМОЩЬЮ НЕКОТОРОГО КЛАССА МАШИН ТЬЮРИНГА

В этом разделе приведены более ранние результаты, полученные автором [8].

Рассмотрим однозначную функцию f одного переменного, которая всегда принимает целое значение $f(x)$ для любого положительного целого значения аргумента x . (В настоящей работе мы ограничиваемся этим классом функций. Рассмотрение более общих классов функций — возможное направление будущих исследований.)

Определим бесконечное множество A натуральных чисел

$$A = \{f(x); x \in N, f(x) > 0\}, \quad (1)$$

где N — множество всех положительных целых чисел. Таким образом, A есть положительная область значений f при целом аргументе x . (Если f — рекурсивная функция, то A — рекурсивно

¹⁾ В годы, последующие за опубликованием данной статьи, появились работы, в которых исследовался вопрос о времени, необходимом для вычисления функций на машинах Тьюринга и других автоматах. — *Прим. ред.*

перечислимое множество.) Мы назовем f **определяющей функцией** множества A и A множеством, определенным f .

Далее определяется метод вычислений в реальное время. Пусть M — автомат с одним двоичным входом и одним двоичным выходом. Пусть $I(t)$ — значение на входе (0 или 1) в момент времени t и $U(t)$ — значение на выходе в момент t , где $t = 1, 2, 3, \dots$. Говорят, что M есть ***C-машина***¹⁾ для A (или для f) тогда и только тогда, когда

$$\begin{aligned} U(t) &= 1, \quad \text{когда} \quad I(t) = 1, \quad \text{а} \quad \left\{ \sum_{i=1}^t I(i) \right\} \in A, \\ U(t) &= 0 \quad \text{в противном случае.} \end{aligned} \quad (2)$$

Другими словами, в момент времени t на выходе стоит 1 в том и только в том случае, если на входе в этот момент стоит 1 и число моментов времени, в которые на входе было 1, является элементом множества A ; это справедливо для любой входной последовательности $I(t)$ и для любого t . Мы говорим также, что A (или f) ***C-вычислимо***, если существует такая машина M . (Заметим, что *C-вычислимо* множество необходимо является рекурсивным, ибо любое рекурсивно перечислимое множество, которое может быть перечислено в порядке возрастания значений, есть рекурсивное множество.) Название «счетчик» [при переводе выбран термин «*C-машина*». — *Ред.*] вызвано тем, что машина считает единицы во входной последовательности $I(t)$, и, как только их число становится элементом множества A , на выходе появляется 1. Так работает наша условная машина. Теперь предположим, что $f(x)$ монотонно возрастает в зависимости от x , и кроме того, $f(x) > 0$ для всех $x > 0$; тогда в *C-машине* для f выход в i -й раз равен 1 в тот момент, когда вход в $f(i)$ -й раз становится равным 1. Когда автомат может работать таким образом в зависимости от f для всех $x > 0$, говорим, что этот автомат является **вычислителем** (computer) **в реальное время**. Конечно, это не единственный способ определить вычислитель в реальное время. При вычислениях в реальное время, определенных таким образом, для того чтобы вычислить $f(x)$, нужно тем больше времени, чем больше x , если f монотонна. Необходимо сохранять скорость каждого логического шага вычислений конечной, как будет показано позднее, когда мы определим структуру M . А пока представьте, что мы определили вычисления в реальное время посредством воспроизведения результатов вычислений, например в виде записи ровно $f(x)$ символов при вычислении $f(x)$ в пределах предписанного конечного времени. То-

¹⁾ В оригинале «counter» — счетчик. Термин *C-машина* выбран потому, что для появляющегося ниже производного термина «countable» не удастся найти хороший эквивалент. — *Прим. перев.*

гда машина M должна записывать по мере возрастания $f(x)$ все бóльшие количества символов за ограниченное время. Но это равносильно требованию, чтобы скорость M становилась произвольно большой по мере возрастания $f(x)$, что невозможно физически.

Поскольку наша интерпретация вычислений в реальное время дана в терминах S -вычислений, мы предполагаем вникнуть в сущность вычислений в реальное время с помощью детального изучения S -машин. Хотя предположения относительно входа S -машин и дают бóльшие возможности, желательно устраивать дело так, чтобы автомат выдавал на выходе в i -й раз 1 в том и только том случае, когда прошло ровно $f(i)$ моментов времени; для этого достаточно подавать на вход последовательность единиц. (В этом случае вход просто служит метрономом.) Грубо говоря, мы хотим, чтобы M была многоленточной машиной Тьюринга с одним двоичным входом и одним двоичным выходом. (Ленты не могут использоваться в качестве входа и выхода) В точных терминах M есть автомат, удовлетворяющий следующим условиям, которые не обязательно являются независимыми друг от друга:

1. Машина M детермирована и имеет один двоичный вход и один двоичный выход.

2. M состоит из управляющего устройства (процессора) P с конечным числом состояний и конечного числа лент. P может писать, стирать и читать символы на лентах и может передвигать ленты вправо или влево. Каждая лента управляется отдельно.

3. Все ленты рассматриваются, как в теории Тьюринга, — они могут расти в оба конца безгранично по мере надобности. Число различных символов, используемых на лентах, конечно.

4. Одна **операция** M состоит либо из записи символов на ленте (в том числе и стирания), либо из передвижения ленты на один квадрат. Существует определенный предел для числа операций, которые машина M может выполнять в единицу времени. (Этот предел определяет скорость работы M .)

5. Входной символ 0 не изменяет («внутреннего») состояния P , не влечет за собой написание или уничтожение символа на ленте, равно как и движение какой бы то ни было ленты. Входной символ 1 вызывает всегда по крайней мере одно из этих действий.

Выше приведено точное определение S -машины M . Здесь нужно заметить, что определенный таким образом класс автоматов есть особый подкласс машин Тьюринга. Они могут иметь более одной растущей ленты. S -машинам позволено начинать работу с лентами, на которые нанесены начальные записи конечной длины. Из условия 4 также следует, что верхняя гра-

ница для емкости памяти машины M в момент времени t есть $p \cdot n \cdot t$, где p — число операций на одной ленте за один момент времени, n — число используемых лент, t — число моментов времени, протекших с начала работы. Таким образом, достижимая емкость памяти растет самое большее линейно относительно времени.

В терминах точного определения S -машины M теперь ясно определение S -вычислений, приведенное выше. Главным содержанием работы [8] было установление классов S -вычислимых функций. Главные результаты, полученные в этой работе, следующие (f, g, h — функции, c и k — положительные целые константы):

1. Если M конечна (что означает, либо что M вообще не имеет лент, либо что M никогда не использует более, чем определенную ограниченную часть каждой из лент), то каждая S -вычислимая функция представима в виде линейного двучлена, за исключением начальных вычислений конечной длины.

2. Если исключить из рассмотрения S -машины, упомянутые в п. 1, то общее состояние S -машины M не может быть одним и тем же в моменты t' и t , где $t \neq t'$, если только входная последовательность между t' и t не состоит из одних нулей. Общее состояние определяется как информация о внутреннем состоянии, записанные на лентах последовательности символов и просматриваемые на каждой ленте квадраты.

3. Любая функция одного переменного, данная в виде полинома с целыми коэффициентами, S -вычислима.

4. Если f и g S -вычислимы, то функции, заданные следующими выражениями, S -вычислимы:

I. $f(x) + g(x)$.

II. $\sum_{i=1}^x f(i)$.

III. $f(x) \cdot g(x)$.

IV. $\prod_{i=1}^x f(i)$.

V. $f(g(x))$.

VI. $f(x)^{g(x)}$.

VII. $f(x) - g(x)$ при условии, что существуют $c > 1$ и k , такие, что $f(x+k) - f(x) \geq \{c/(c-1)\} \cdot [g(x+k) - g(x)]$.

5. Функция, определенная следующим образом, S -вычислима:

$$h(1) = c,$$

$$h(x+1) = c^{h(x)}.$$

6. Если $\sum_{i=1}^x f(i)$ C -вычислима и $f(x+k) \geq f(x) + \frac{1}{c-1} \sum_{i=1}^x x^{k-1} f(i)$

для некоторых $c > 1$ и k , то C -вычислима и f .

7. Если функция двух переменных h вводится как определяющая функция множества A , где $A = \{h(x, y) \mid x, y, h(x, y) \text{ — натуральные числа}\}$ и если f и g C -вычислимы, то C -вычислимы следующие функции:

I. $h(x, y) = f(x) + f(y)$, если для некоторого k $f(x+k) \geq \geq 2f(x)$;

II. $h(x, y) = f(x+y) + g(x)$, если для некоторого k

$$f(x+k+1) - f(x+1) > g(x) - g(1);$$

8. Если A и B — C -вычисляемые множества, то $A \cup B$, $A \cap B$ и $\sim A$ также C -вычислимы.

9. Любая C -машина (из класса, определенного выше) с лентами, которые могут расти в оба конца, имеет эквивалентную C -машину с полуограниченными лентами.

10. Любая C -машина с n лентами имеет эквивалентную C -машину только с 2^n внутренними состояниями.

11. Любая C -машина имеет эквивалентную C -машину всего с двумя символами.

Доказательства всех этих результатов основаны на построении C -машин для решения задач. Большинство из них связано с использованием свойств абстрактных логических сетей. Построение C -машины для полиномиального множества, приведенное в следующем разделе, дает пример такого доказательства. Это построение довольно прозрачно по сравнению с теми, которые использованы для доказательства других результатов [8].

3. C -ВЫЧИСЛИМОСТЬ ПОЛИНОМИАЛЬНЫХ МНОЖЕСТВ

Пусть p — полином n -й степени с целыми коэффициентами

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0,$$

где $a_n > 0$. Тогда **полиномиальное множество** $A(p)$ определяется как бесконечное множество, состоящее из элементов $p(0)$, $p(1)$, $p(2)$, $\dots$.

Например, если мы возьмем полином $p(x) = x^2$, то $A(p)$ будет множеством всех точных квадратов. (Легко проверить, что единственными полиномиальными множествами, C -вычислимыми на конечных автоматах, являются множества, определяемые линей-

ными полиномами. Более того, все линейные полиномы S -вычислимы таким образом.)

Теорема 1. Каждое полиномиальное множество является S -вычислимым.

Доказательство. Доказательством будет являться построение S -машины для данного множества. В силу основного свойства полинома существует k_0 , такое, что для всех $x > k_0$ полиномы $p^{(i)}$ монотонно возрастают, где $p^{(i)}$ — разности, определяемые ниже. Для такого числа k_0 строится следующая таблица разностей:

$$\begin{array}{ccccccc}
 & p(k_0) & & & & & \\
 & \swarrow & & \searrow & & & \\
 p(k_0+1) & & p'(k_0+1) & & & & \\
 & \swarrow & & \searrow & & & \\
 p(k_0+2) & & p'(k_0+2) & & p''(k_0+2) & & \\
 & \swarrow & & \searrow & & \dots & \\
 p(k_0+3) & & p'(k_0+3) & & p''(k_0+3) & & p^{(n)}(k_0+n) \\
 & \swarrow & & \searrow & & \dots & \\
 p(k_0+4) & & p'(k_0+4) & & p''(k_0+4) & & p^{(n)}(k_0+n+1) \\
 & \swarrow & & \searrow & & \dots & \\
 p(k_0+5) & & p'(k_0+5) & & p''(k_0+5) & & p^{(n)}(k_0+n+2) \\
 & \vdots & & \vdots & & \vdots & \\
 & \vdots & & \vdots & & \vdots & \\
 & \vdots & & \vdots & & \vdots &
 \end{array}$$

где

$$\left. \begin{array}{l}
 p'(r) = p(r) - p(r-1), \\
 p''(r) = p'(r) - p'(r-1), \\
 \dots \\
 p^{(n)}(r) = p^{(n-1)}(r) - p^{(n-1)}(r-1).
 \end{array} \right\} \quad (3)$$

Хорошо известно, что $p^{(n)}(r) = n!a_n$ для всех r . Другими словами, p можно характеризовать с помощью $p(k_0)$, $p'(k_0+1)$, ..., $p^{(n)}(k_0+n)$, а не a_n , a_{n-1} , ..., a_0 . Более того, легко видеть, что существует $k \geq k_0$, такое, что условие

$$p'(r+n) > p''(r+n) + p'''(r+n) + \dots + p^{(n)}(r+n) \quad (4)$$

выполнено для всех $r > k$ в силу того факта, что p' есть полином $(n-1)$ -й степени и все полиномы в правой части (4) имеют меньшие степени. Взяв это k вместо k_0 в вышеприведенной

таблице разностей, мы строим для $A(p)$ С-машину с n лентами:

$$\begin{array}{rcl}
 & \overbrace{1 \ 1 \ \dots \ 1}^{p'(k+n)} & \\
 T_1 & \frac{\quad}{\quad} & \\
 & \uparrow & \\
 & \overbrace{1 \ 1 \ \dots \ 1}^{p''(k+n)-1} & \\
 T_2 & \frac{\quad}{\quad} & \\
 & \uparrow & \\
 & \overbrace{1 \ 1 \ \dots \ 1}^{p'''(k+n)-1} & \\
 T_3 & \frac{\quad}{\quad} & \\
 & \uparrow & \\
 \vdots & & \vdots \\
 & & \\
 & \overbrace{1 \ 1 \ \dots \ 1}^{p^{(n)}(k+n+1)-1} & \\
 T_n & \frac{\quad}{\quad} & \\
 & \uparrow &
 \end{array}$$

Стрелки под лентами обозначают символы, которые считывает машина. Употребляя эти ленты, машина выполняет в те моменты времени, когда на вход поступает символ 1, следующие операции:

1. Один шаг вправо в каждый момент на T_1 .
2. Одновременно с 1 продвигается вправо на T_n и влево на T_{n-1} , записывая 1 на T_{n-1} до тех пор, пока не будут исчерпаны все 1 на T_n . В следующий момент пишет 1 на T_{n-1} .
3. После 2, но все еще одновременно с 1 продвигается вправо на T_{n-1} и влево на T_{n-2} , записывая 1 до тех пор, пока все 1 на T_{n-1} не будут исчерпаны. В следующий момент пишет 1 на T_{n-2} .
4. Повторяет процедуру с (T_{n-2}, T_{n-3}) , (T_{n-3}, T_{n-4}) , ..., (T_3, T_2) , все еще совпадающую с 1. (В силу (4) на T_1 еще есть 1.)
5. После этих шагов все еще продолжает 1, пока все 1 на T_1 не будут исчерпаны.
6. Затем продвигается вправо на T_2 и T_1 , записывая 1 на T_1 до тех пор, пока все 1 на T_2 не будут исчерпаны. После этого пишет 1 на T_1 , выдает на выходе 1, продвигается на один шаг на $T_2, T_3, \dots, T_n$ влево и считывает крайние правые символы на лентах.

Когда вышеприведенный процесс закончен, ленты в силу (3) выглядят следующим образом:

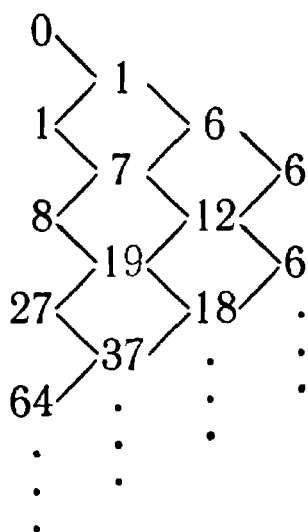
$$\begin{array}{rcc}
 & \overbrace{p'(k+n+1)} & \\
 T_1 & \frac{1 \ 1 \ \dots \ 1}{} & \uparrow \\
 & \overbrace{p''(k+n+1)-1} & \\
 T_2 & \frac{1 \ 1 \ \dots \ 1}{} & \uparrow \\
 & \overbrace{p'''(k+n+1)-1} & \\
 T_3 & \frac{1 \ 1 \ \dots \ 1}{} & \uparrow \\
 & \vdots & \\
 & \vdots & \\
 & \vdots & \\
 & p^{(n)}(k+n+2)-1 = p^{(n)}(k+n+1)-1 & \\
 T_n & \frac{1 \ 1 \ \dots \ 1}{} & \uparrow
 \end{array}$$

Для каждой ленты T_i здесь существенно использовано равенство $p'(r) = p'(r-1) + p''(r-1) + \dots + p^{(n)}(r-1)$. Легко проверить, что общее число единиц на входе, требуемых, чтобы выполнить описанные операции, есть $p'(k+n+1)$. Это в точности приращение нашего полинома p от $p(k+n)$ до $p(k+n+1)$. Следовательно, если мы устроим машину так, что в момент подачи $p(k+n)$ -го входного символа 1 будут выполнены описанные выше начальные условия для лент, то следующие входные единицы приведут к вышеописанным операциям, по окончании которых мы сосчитаем значение $p(k+n+1)$.

Установление начальных условий для лент и отсчет $p(k+n)$ входных символов являются конечными операциями, которые могут быть выполнены конечным автоматом. Когда вычислено $p(k+n+1)$, ленты имеют вид, сходный с начальным с той разницей, что количество единиц на каждой ленте изменилось в соответствии с изменившимся на 1 аргументом функций. Кроме того, обозреваются символы на правых концах лент, а не на левых, как это было при начальном расположении. Следовательно, если мы снова начнем производить все те же операции с лентами, но выполняя все передвижения в противоположном направлении, мы по их окончании сосчитаем $p(k+n+2)$. Затем машина повторяет те же операции, сдвигая головки назад и вперед и руководствуясь одним из двух наборов правил, приведенных

выше. В результате машина может вычислять $p(k+n+3)$, $p(k+n+4)$ и т. д. до бесконечности. Таким образом, у нас есть вычисление функции p , начиная со значения аргумента $k+n+1$. Осталось показать, как вычисляется p для значений аргумента между 1 и $k+n$. Поскольку вычисление (положительных) значений от $p(1)$ до $p(k+n)$ выполняется с помощью конечного числа операций, добавление конечного автомата решит вопрос. Действительно, эти вычисления выполняются подачей выходного символа 1 в подходящие моменты начального отрезка времени вплоть до $p(k+n)$ момента. Таким образом, мы показали принцип S -вычисления функции p посредством простого описания механического поведения машины. Ее поведение может также быть строго описано в терминах (растущего) графа состояний. Доказательство окончено.

В качестве частного примера построим S -машину, которая вычисляет $A = \{n^3; n=0, 1, 2, \dots\}$ для входного символа 1. Таблица разностей в этом случае такая:



Поскольку $6 + 12 < 19$, равенства $p_3'''(k+3)=6$, $p_3''(k+3)=12$ и $p_3'(k+3)=19$ удовлетворяют (4). Именно мы можем выбрать начальный вид лент следующим образом:

$$\begin{array}{rcl}
 & & \overbrace{19} \\
 T_1 & \overbrace{1 \ 1 \ \dots \ 1} & \\
 & \uparrow & \\
 & & \overbrace{11} \\
 T_2 & \overbrace{1 \ 1 \ \dots \ 1} & \\
 & \uparrow & \\
 & & \overbrace{5} \\
 T_3 & \overbrace{1 \ 1 \ 1 \ 1 \ 1} & \\
 & \uparrow &
 \end{array}$$

Поведение C -машин можно описать в терминах множества N -ок, что является легкой модификацией множества пятерок, использованного Тьюрингом в [2]. Хотя N -ки имеют различные формы, типичный их вид следующий:

$$\{q_i, S_{i_1}, \dots, S_{i_n}, S_{j_1}, \dots, S_{j_n}, M_{k_1}, \dots, M_{k_n}, q_j, U\},$$

где

- n — число лент машины,
- q_i — текущее состояние процессора P ,
- $S_{i_1}, \dots, S_{i_n}$ — наблюдаемые в данный момент на лентах $T_1, \dots, T_n$ символы,
- $S_{j_1}, \dots, S_{j_n}$ — новые символы, которые нужно записать на лентах $T_1, \dots, T_n$,
- $M_{k_1}, \dots, M_{k_n}$ — движения по лентам $T_1, \dots, T_n$,
- q_j — следующее состояние P ,
- U — выходной символ.

Теперь представим полиномиальную C -машину в терминах этих N -ок. Входной символ 0 не оказывает на машину никакого действия, а вход 1 вызывает следующие операции. Звездочки на месте символов ленты означают, что остаются написанные на лентах символы (или что записываются идентичные им символы) L , R и $*$ на месте движений означают сдвиг на единицу влево, вправо и отсутствие движения соответственно.

1. q_0 111, ***, *** q_1 1
2. q_1 111, ***, *** q_2 0
- ...
8. q_7 111, ***, *** q_8 1
9. q_8 111, ***, *** q_9 0
- ...
26. q_{25} 111, ***, *** q_{26} 0
27. q_{26} 111, ***, *** q_{27} 1

(Это и есть начальный сегмент; отметьте строки 1 ($=1^3$), 8 ($=2^3$), 27 ($=3^3$), которые имеют выход 1.)

28. q_{27} 111, ***, RLR q_{28} 0
29. q_{28} 101, $*1*$, RLR q_{28} 0
30. q_{28} 100, $*1*$, $R**$ q_{29} 0
31. q_{29} 100, ***, $R**$ q_{29} 0
32. q_{29} 010, $1**$, $RR*$ q_{30} 0
33. q_{30} 010, $1**$, $RR*$ q_{30} 0
34. q_{30} 000, 100, $*LL$ q_{31} 1

Таким образом, следующая выходная единица появляется в момент 64 ($=4^3$), а ленты выглядят в этот момент следующим образом:

$$\begin{array}{rcc}
 & \overbrace{\quad\quad\quad}^{37} & \\
 T_1 & \frac{1 \ 1 \ \dots \ 1}{\quad} & \uparrow \\
 & \overbrace{\quad\quad\quad}^{17} & \\
 T_2 & \frac{1 \ 1 \ \dots \ 1}{\quad} & \uparrow \\
 & \overbrace{\quad\quad\quad}^5 & \\
 T_3 & \frac{1 \ 1 \ 1 \ 1 \ 1}{\quad} & \uparrow
 \end{array}$$

Следующие шаги являются зеркальным отражением вышеприведенных шагов относительно движения:

- 35. q_{31} 111, ***, LRL q_{32} 0
- 36. q_{32} 101, *1*, LRL q_{32} 0
- 37. q_{32} 100, ***, $L**$ q_{33} 0
- 38. q_{33} 100, ***, $L**$ q_{33} 0
- 39. q_{33} 010, 1**, $LL*$ q_{34} 0
- 40. q_{34} 010, 1**, $LL*$ q_{34} 0
- 41. q_{34} 000, 100, * RR q_{27} 1

Затем весь процесс повторится от q_{27} до q_{34} . Это конец построения C -машины для точных кубов.

Нужно отметить, что запись на T_3 не растет во времени. Это верно и в общем случае полинома n -й степени, только записи на $n-1$ ленте удлиняются. Другим моментом, который мы должны здесь отметить, является то, что, если дана определяющая функция f , задача вычислений сводится к последовательному вычислению $f(1)-f(0)$, $f(2)-f(1)$, $f(3)-f(2)$ и т. д. Другими словами, если мы определяем приращение F функции f как

$$F(x) = f(x+1) - f(x), \quad x > 0,$$

то перед машиной стоит задача вычислять $F(x)$ заново для каждого $x=1, 2, 3$ и т. д. Это просто следует из того факта, что поскольку вычислено $f(x)$, то и $f(x+1)$ во многом вычислено, и чтобы получить $f(x+1)$, требуется единственное дополнительное вычисление — вычисление разности $f(x+1) - f(x)$.

4. РЕКУРСИВНЫЕ ФУНКЦИИ, НЕ ЯВЛЯЮЩИЕСЯ С-ВЫЧИСЛИМЫМИ В РЕАЛЬНОЕ ВРЕМЯ

По ходу рассуждений, приведенных в предыдущих разделах, встает вопрос, являются ли решительно все рекурсивные функции вычислимыми описанным образом. Легко поверить, что не все рекурсивные функции S -вычислимы, ибо условие реального времени накладывает на действия машины Тьюринга по-настоящему строгое ограничение. Однако доказательство отсутствовало. В настоящем разделе мы устанавливаем существование рекурсивных функций, которые не являются S -вычислимыми.

Сначала докажем предварительную теорему о S -машинах.

Теорема 2. *Если функция f S -вычислима S -машиной, у которой до начала вычислений на лентах записаны слова конечной длины, то существует эквивалентная S -машина с первоначально пустыми лентами.*

Доказательство. Доказательство довольно простое. Независимо от того, какие (непустые) слова записаны первоначально на лентах у исходной машины, в процессе вычислений они будут просмотрены машиной хотя бы один раз спустя конечное число шагов с начала вычислений. В противном случае некоторые части слов никогда не будут использованы, и их можно стереть перед началом вычислений.

Пусть m — наименьшее число операций, которое требуется головкам машины, чтобы просмотреть все символы на лентах по крайней мере по одному разу. Пусть также n — наибольшее из расстояний между двумя любыми символами на любой ленте к концу просмотра начальных слов так, как это описано выше. Очевидно, $m \geq n$. Также очевидно, что поведение машины во время первых m шагов есть событие с конечным числом состояний и его можно моделировать конечным автоматом. Слова, которые записаны на лентах через m тактов работы, можно получить за n тактов, если начинать вычисления с пустыми лентами. Следовательно, на начальном отрезке действий исходную машину можно заменить описанными выше моделирующей машиной и автоматом, создающим содержание лент. Получившаяся S -машина эквивалентна исходной, как и утверждает теорема.

Здесь мы напоминаем читателям, что существует конечный алгоритм для нумерации всех S -машин, представленных в виде множества наборов длины N (N -ок), как это обсуждалось в предыдущем разделе, или в другом подобном виде [2, 3]. Эта процедура есть точный аналог нумерации всех машин Тьюринга с помощью гёделевских номеров. Поэтому мы не вдаемся

в детали. Знание этого факта позволяет перейти к доказательству следующей теоремы.

Теорема 3. *Существует счетный бесконечный класс рекурсивных функций, не вычислимых в нашем специальном классе машин Тьюринга (С-машин).*

Доказательство. Доказательство основано на «диагональном» методе. Без потери общности предполагаем в доказательстве, что входная последовательность любой С-машины всегда состоит только из единиц. Пусть $C_1, C_2, \dots$ — последовательность С-машин, в которой C_i либо С-машина с гёделевским номером i , либо некоторая С-машина, на входе которой всегда 0, если i не является гёделевским номером ни для какой С-машины. Поскольку каждой С-машине с первоначально пустыми лентами соответствует некоторый гёделевский номер, она, очевидно, должна появиться среди машин $C_1, C_2, \dots$, которые занумерованы описанным образом. В дополнение к этому теорема 2 устанавливает, что для любой С-машины с первоначально непустыми лентами существует по крайней мере одна эквивалентная С-машина с первоначально пустыми лентами. Следовательно, все С-машины, определенные нами, представлены в нумерации.

Теперь предположим, что существует С-машина K со следующим выходом. В момент t выход K есть отрицание выхода машины C_t в момент t . Выход K отличается от выхода любой С-машины из всей приведенной последовательности по крайней мере в один момент времени. Поэтому K или другая С-машина с таким же выходом не может быть в нумерации всех С-машин. Но мы знаем, что наша нумерация содержит все возможные С-машины. Следовательно, С-машина K не существует. Вопрос о том, является или нет n номером набора символов, решается алгоритмически. Затем решить вопрос о том, имеет ли последовательность с номером n форму, представляющую С-машину в нашем определении, можно с помощью рекурсивной процедуры. Далее, если последовательность символов представляет С-машину, мы можем заставить эту машину начать работу с пустыми лентами и проработать n тактов, а затем посмотреть выход в n -й момент. Это рекурсивная процедура. Наконец, отрицание выхода также является рекурсивной процедурой. Следовательно, композиция всех этих процедур — рекурсивная процедура. Более точно, множество S всех t , таких, что выход K в момент t есть 1, является рекурсивным; но, так как K не содержится среди С-машин, это множество С-невычислимо. Определим теперь функцию f следующим образом: $f(1)$ — наименьший элемент S , $f(x+1)$ есть наименьший элемент множества S , от-

личный от всех $f(1), f(2), \dots, f(x)$. Поскольку S рекурсивно, то и f рекурсивно. Но, если бы f была S -вычислимой функцией, S было бы S -вычислимым множеством. Следовательно, f — рекурсивная S -невычислимая функция.

Поскольку мы получили S -невычислимую функцию, можно построить счетное бесконечное семейство S -невычислимых рекурсивных функций, употребляя m как параметр следующим образом:

$$f_m(x) = f(x) + g(m),$$

где g — монотонно возрастающая функция, которая принимает целые значения при целых значениях m . Простейший пример — $g(m) = m$. Можно показать, используя теорему, сформулированную в п. 4) (I) разд. 2, что f_m , полученные таким образом, S -невычислимы, если f S -невычислима. Заметим, что множества, определяемые семейством f_m , не пересекаются.

Мы использовали здесь метод доказательства, похожий на тот, которым доказывают, что существуют функции, не являющиеся рекурсивными. Однако имеется существенное различие. В доказательстве упомянутой теоремы мы утверждаем, что, поскольку функция, построенная с помощью диагонального процесса, не может быть представлена никакой машиной Тьюринга из занумерованных, такая нумерация всех машин Тьюринга невозможна. Следовательно, не существует рекурсивной функции f , такой, что $f(x) = 1$, если x — гёделевский номер машины Тьюринга, представляющей рекурсивную функцию, и $f(x) = 0$ в противном случае. Это проистекает из того факта, что мы не всегда можем решить за конечное число шагов, является ли x гёделевским номером машины Тьюринга, представляющей рекурсивную функцию. Другими словами, если мы в течение n шагов вычислений проверяем, дает ли машина выход для данного входа, для машины всегда существует возможность давать выход на $(n+1)$ -м шаге или позже, сколь большим ни взять n . В доказательстве нашей теоремы мы были свободны от этой трудности, потому что должны были проверить выход S -машины S_k только при $t = k$, что является конечным процессом.

Вышеприведенная теорема доказана в связи с классом S -машин, определенных в разд. 2. Однако теорема может быть далее обобщена следующим образом.

Теорема 4. Независимо от того, каковы конструкция и операции машины, если только они рекурсивно определены и результат вычислений определяется числом единиц, поданных на вход к моменту данного выхода, существует бесконечное счетное

множество рекурсивных функций, которые не вычислимы в классе машин, определенных таким образом.

Доказательство. Если мы внимательно просмотрим доказательство теоремы 3, то увидим, что нам не понадобились иные условия, накладываемые на структуру машины, кроме данных в теореме. Детали доказательства оставляем читателям.

Теорема 4 в действительности является следствием теоремы 3, или наоборот. Однако машины, используемые в теореме 4, сильно отличаются от машин из теоремы 3, и по этой причине мы приводим теоремы отдельно. В качестве примеров машин этого класса упомянем следующие:

1. Мы можем начинать работу с непустыми лентами.
2. Могут быть машины, у которых ленты имеют более, чем по одной рабочей головке.
3. Переменная скорость работы может быть определена рекурсивной функцией (если не считаться с физической возможностью реализации).

Легко сформулировать другие примеры.

ЗАКЛЮЧЕНИЕ

Мы рассмотрели некоторые результаты, связанные с вычислениями в реальное время. Однако автор чувствует, что это не более, чем начальная попытка. То установление связей между входом и выходом, которое мы использовали, ни в коем случае не является единственно возможным¹⁾. Исследования должны вестись также для такого устройства: результат вычисления $f(k)$ дается в виде содержания лент в момент, когда подается $C(k)$ -й вход, где C — некоторая функция k . Другие связанные с этим проблемы предлагаются в исследовании [8].

Настоящую попытку можно расценить как стремление развить теорию машин Тьюринга с ограничениями, наложенными на число шагов вычислений. Это можно противопоставить изучению теории машин Тьюринга с различными ограничениями на длину лент, предпринятому Майхиллом [5], Рабином, Скоттом [6], Шепердсоном [7], Ричи [9] и другими.

Нужно также заметить, что после того, как настоящая работа была написана, Зигель [10] получил, в частности, такие результаты:

- 1) не каждая C -вычислимая функция примитивно рекурсивна;
- 2) не каждая монотонно возрастающая примитивно рекурсивная функция C -вычислима.

¹⁾ Дальнейшие исследования в этом направлении проводились в [11, 13] и других работах. — *Прим. ред.*

Машины, которые он использовал для классификации рекурсивных функций, определены в разд. 2. Дальнейшее обобщение классификации, основанное на машинах, используемых в теореме 4, определяет возможное направление будущих исследований.

ЛИТЕРАТУРА

1. McNaughton R., Theory of automata, a survey, *Advances in Computers*, vol 2, F. L. Alt, Ed., Academic Press, Inc., N. Y., 1961.
2. Turing A. M., On computable numbers, with an application to the Entscheidungsproblem, *Proc. London Math. Soc.*, ser. 2, **42** (1936), 236—265. («Correction», ser. 2, **43** (1937), 544—546.)
3. Davis M., Computability and unsolvability, McGraw-Hill Book Co., Inc., N. Y., 1958.
4. Lee C. Y., Various models and examples of automata, in *Programming concepts, automata, and adaptive systems*, 1960 Summer Session, College of Engrg., Univ. of Michigan, Ann Arbor.
5. Myhill J., Linear bounded automata, WADD Tech. Note 60—165 and University of Pennsylvania, Philadelphia, Moore School Rept. № 60-22, June 1960.
6. Rabin M. O., Scott D., Finite automata and their decision problems, *IBM J. Res. and Dev.*, **3** (1959), 114—125. (Русский перевод: Кибернетический сборник, вып. 4, ИЛ, М., 1962.)
7. Shepherdson J. C., The reduction of two-way automata to one-way automata, *IBM J. Res. and Dev.*, **3** (1959), 198—200. (Русский перевод: Кибернетический сборник, вып. 4, ИЛ, М., 1962.)
8. Yamada H., Counting by a class of growing automata, Moore School of Electrical Engrg.; University of Pennsylvania, Ph. D. dissertation, June 1960.
9. Ritchie R. W., Classes of recursive functions of predictable complexity, Princeton University, Princeton, N. J., Ph. D. dissertation, 1961. (См. также стр. 50—93 настоящего сборника.)
10. Siegel J., Some theorems about Yamada's restricted class of recursive functions, IBM Thomas J. Watson Research Center, Yorktown Hgts., N. Y., IBM Res. Paper RC-510; 1961.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

11. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, **117** (1965), 285—306. (Русский перевод: Хартманис Дж., Стирнз Р. Е., О вычислительной сложности алгоритмов, Кибернетический сборник (новая серия), вып. 4, «Мир», М., 1967, стр. 57—85.)
12. Трахтенброт Б. А., Тьюринговы вычисления с логарифмическим замедлением, *Алгебра и логика, Семинар*, **3** (1964), 33—48.
13. Rabin M. O., Real-time computation, *Israel J. of Math.*, **1**, № 4 (1963), 203—211. (См. стр. 156—167 настоящего сборника.)

ВЫЧИСЛЕНИЯ В РЕАЛЬНОЕ ВРЕМЯ¹⁾

М. Рабин

При формулировке задачи вычислений на машинах Тьюринга предполагается, что исходные данные (скажем, числовое значение аргумента, для которого нужно вычислить значение функции) записаны на ленте машины и машина работает, выполняя вычисления. На число шагов («время»), необходимых для того, чтобы закончить вычисления, не накладывается никаких предварительных ограничений. Функции, вычисляемые таким образом, — это в точности все рекурсивные функции.

С точки зрения общей теории вычислений весьма интересно проникнуть в сущность вычислительных процессов, при которых на время вычислений наложены некоторые ограничения.

Одно естественное ограничение заключается в том, что если входные данные состоят из n символов, то требуется выполнить вычисления за n основных шагов — один шаг на один входной символ. Мы можем предположить, что входная последовательность поступает в машину по одному символу за единицу времени и что машина выполняет одну элементарную операцию на один выходной символ. Мы по-прежнему будем считать наши машины машинами Тьюринга, которые, однако, могут иметь более одной ленты. Вычисления, которые выполняются таким образом, будут называться **вычислениями в реальное время** (на машине Тьюринга). Отметим, что наши системы могут служить математическими моделями для вычислительных машин (с внешней памятью на лентах), которые используются для так называемого управления «в реальное время».

В частности, если результат вычислений для любого входа 0 или 1, то можно считать, что машина определяет некоторое множество, именно множество тех входных последовательностей, которые дают 1. Неформально мы можем также сказать, что машина распознает, содержится данная входная последовательность в множестве, определенном машиной, или нет.

Имеется несколько результатов (наиболее значительные у Ямады [2]) о вычислениях в реальное время на машинах Тью-

¹⁾ Rabin M., Real time computation, *Israel J. of Math.*, 1, № 4 (1963), 203—211.

ринга. Они идут главным образом по линии установления возможности определенных вычислений в реальное время. Понятие вычислений в реальное время интересно обобщается Хартманисом и Стирнзом [1].

В отличие от случая конечных автоматов здесь нет четкой внутренней характеристики класса множеств, которые определены в реальное время. Действительно, прежде чем давать полную характеристику, мы, вероятно, должны приобрести некоторую интуицию относительно осуществимости и неосуществимости определенных задач в реальное время.

Наш основной результат состоит в том, что существует проблема распознавания, которая может быть решена в реальное время, если использовать две ленты, и не может быть решена в реальное время, если использовать только одну ленту. Результат имеет очевидные приложения, касающиеся вопроса о сравнительных возможностях вычислительных машин с одной или большим количеством лент при использовании их в устройствах управления в реальное время.

В разд. 6 мы обсуждаем различие в трудности доказательств невозможности некоторых вычислений. В качестве иллюстрации дается пример задачи, которая неожиданным образом может быть решена в реальное время на одной ленте (теорема 3).

Результат об относительных возможностях вычислений на одной ленте в сравнении с вычислениями на двух лентах — лишь пример из трудной области оценки «степени сложности вычислений». Мы надеемся, что некоторые идеи нашего доказательства, в особенности понятие *узкого места* (bottleneck) в вычислениях, можно распространить применительно к другим задачам из этой же области.

1. МАШИНЫ ТЬЮРИНГА, РАБОТАЮЩИЕ В РЕАЛЬНОЕ ВРЕМЯ

Модель вычислений в реальное время, с которой мы работаем, такая же, как у Хартманиса и Стирнза [1] и у Ямады [2].

Многоленточная машина Тьюринга во *входном алфавите* Σ есть конечный автомат M , у которого имеется конечное множество S *состояний* и *рабочий алфавит* $W = \{\alpha, \dots, \alpha_n\}$. Одно из состояний (назовем его s_0) мы отличаем от других как *начальное состояние*. Подмножество $F \subseteq S$ выбирается в качестве множества *выделенных конечных состояний*. Машина имеет k двусторонних бесконечных линейных *рабочих лент* $t_1, \dots, t_k$, которые разделены на квадраты. Имеется головка, способная читать и писать и воспринимающая в любой данный момент один квадрат на каждой из рабочих лент. M может принимать на входе $\sigma \in \Sigma$. Мы всегда предполагаем, что рабочий алфавит

содержит пустой символ и по крайней мере еще один символ, так что $2 \leq n$.

Работа машины определяется функцией

$$M(\sigma, s, \alpha_{i_1}, \dots, \alpha_{i_k}) = (s', X_1, \dots, X_k, \alpha_{j_1}, \dots, \alpha_{j_k}),$$

где $\sigma \in \Sigma$, $s, s' \in S$, $\alpha_i, \alpha_j \in W$, $X_i \in \{0, 1, -1\}$. Мы будем говорить об этой функции как о **таблице** k -ленточной машины Тьюринга. Интерпретация состоит в том, что если на входе σ и M в состоянии s читает α_{i_r} на ленте t_r , $1 \leq r \leq k$, то M перейдет в состояние s' , напишет α_{j_r} в квадрате, который она воспринимает на ленте t_r , и сдвинет каждую ленту t_r на один квадрат влево, вправо или не сдвинет вовсе, смотря по тому, равно ли X_r 1, -1 или 0. Это действие M называется **элементарной операцией**.

Замечание. Когда M пишет символ $\alpha \in W$ в некотором квадрате, содержимое этого квадрата предварительно стирается. Написание пустого символа из W означает просто стирание содержимого квадрата.

Множество всех конечных последовательностей в алфавите Σ будет обозначаться Σ^* .

Определение 1. О последовательности $x = \sigma_1 \dots \sigma_p \in \Sigma^*$ говорят, что она **допускается** машиной M , если из состояния s_0 при пустых рабочих лентах M под действием последовательности x при помощи элементарных операций закончит свою работу в состоянии, принадлежащем F (т. е. в момент времени p машина M находится в выделенном состоянии).

Множество всех последовательностей, допустимых машиной M , называют множеством, **определимым** M , и обозначают $T(M)$. Множество $T \subseteq \Sigma^*$ называют **определимым в реальное время** (или **распознаваемым**), если существует многоленточная машина Тьюринга M , такая, что $T = T(M)$.

В частности, T называют **определимым в реальное время с помощью k лент**, если для некоторой машины M с k лентами $T(M) = T$.

Совершенно ясно, что термин «реальное время» хорошо подходит для такого способа работы. Если x — последовательность длины p — вводится в машину M за p единиц времени, то посмотрев в момент времени p на состояние машины, мы узнаем, допустима или нет последовательность x . Таким образом, между восприятием исходных данных и их переработкой нет задержки во времени.

Мы ограничиваемся рассмотрением проблем распознавания. Простое исследование показывает, однако, что проблема вычис-

лений в реальное время может быть легко сведена к проблеме распознавания. Таким образом, наши ограничения не влекут потерю общности.

2. МНОЖЕСТВО T_2

Пусть $\Sigma = \{a, b, 0, 1, \alpha, \beta\}$. Слова из символов $\{a, b\}$ будем называть *ab-словами* и множество *ab-слов* обозначать через A . Слова в алфавите $\{0, 1\}$ будем называть *01-словами* и множество их обозначать через Z .

Если $x = \sigma_1 \sigma_2 \dots \sigma_{n-1} \sigma_n$, то по определению $x^* = \sigma_n \sigma_{n-1} \dots \sigma_2 \sigma_1$.

Определение 2.

$$T_2 = \{uvau^* \mid u \in A, v \in Z\} \cup \{uv\beta v^* \mid u \in A, v \in Z\}.$$

Лемма 1. *Множество T_2 определимо в реальное время двуленточной машиной.*

Доказательство. Мы опишем работу двуленточной машины M , для которой $T_2 = T(M)$. Читатель может проверить, что такой способ работы действительно реализуется в виде некоторой таблицы машины.

По мере поступления *ab-слова* u M будет записывать его на своей первой ленте. Когда будет поступать *01-слово* v , M будет записывать его на своей второй ленте. В зависимости от того, какой из символов α или β последует за входным словом uv , M будет просматривать в обратном направлении первую или вторую ленту. M окончит работу в выделенном состоянии тогда и только тогда, когда последовательность w , идущая вслед за α (или β), совпадет с последовательностью, которая получается просмотром в обратном направлении первой (второй) ленты.

Теорема 1. *Множество T_2 не определимо в реальное время одноленточной машиной. Следовательно, вычисления с двумя лентами в реальное время обладают большими возможностями, чем вычисления в реальное время с одной лентой.*

3. ПРЕДВАРИТЕЛЬНЫЕ ЛЕММЫ

Чтобы доказать, что T_2 не определимо в реальное время одноленточной машиной, предположим противное, именно что некоторая одноленточная машина M определяет T_2 в реальное время. Пусть число состояний M будет t и число букв в рабочем алфавите будет n . На протяжении последующих разделов

(3—5) M всегда будет обозначать эту фиксированную одноленточную машину, для которой $T_2 = T(M)$.

Определение 3. Если w — входная последовательность машины M , то *рабочей зоной* $t(w)$ машины M при последовательности w назовем последовательность квадратов ленты, на которых побывала машина M во время подачи входа w .

Если x — последовательность квадратов на ленте или последовательность символов, то $l(x)$ будет обозначать *длину*, т. е. число элементов x .

Пусть x — входная последовательность; *кодом* x мы будем считать последовательность символов на квадратах рабочей зоны $t(x)$, состояние машины M и ее положение на ленте в конце подачи входной последовательности x .

Лемма 2. Существует числовая константа $c > 0$, такая, что для любого $u \in A$ и любого целого $i > 0$ существует $v \in Z$, такое, что $l(v) = i$ и $ci \leq l(t(uv))$.

Доказательство. Существует 2^i последовательностей $v \in Z$, таких, что $l(v) = i$. Поскольку после uv на входе может появиться β , то, если $v_1 \neq v$, тогда uv_1 и uv должны иметь разные коды. В противном случае и $uv\beta v^*$ и $uv_1\beta v^*$ будут допускаться машиной M .

Пусть $l(t(uv)) \leq k$ для всех $v \in Z$, $l(v) = i$. Тогда существует самое большее $n^k \cdot k \cdot t$ различных кодов для входов uv . Следовательно, $2^i \leq n^k \cdot k \cdot t$. Если i велико, то и k велико, так что можно предполагать $kt \leq n^k$ (мы предполагаем $2 \leq n$). Таким образом, $2^i \leq n^{2k}$, и, следовательно,

$$\frac{1}{2} \frac{\ln 2}{\ln n} i \leq k.$$

Мы можем взять $c_1 = \frac{1}{2} \frac{\ln 2}{\ln n}$. Это c_1 будет пригодно для всех i , больших, чем некоторое i_0 ; для достаточно малого c лемма будет выполняться при всех i .

Лемма 3. Существует целое $d > 0$ (зависящее только от машины M), такое, что для любого $u \in A$ и любого целого $i \geq l(u)$ существует последовательность $v \in Z$, $l(v) = i$, такая, что а) $ci \leq l(t(uv))$, б) не более чем $1/5$ квадратов рабочей зоны $t(uv)$ посещались машиной M более чем d раз.

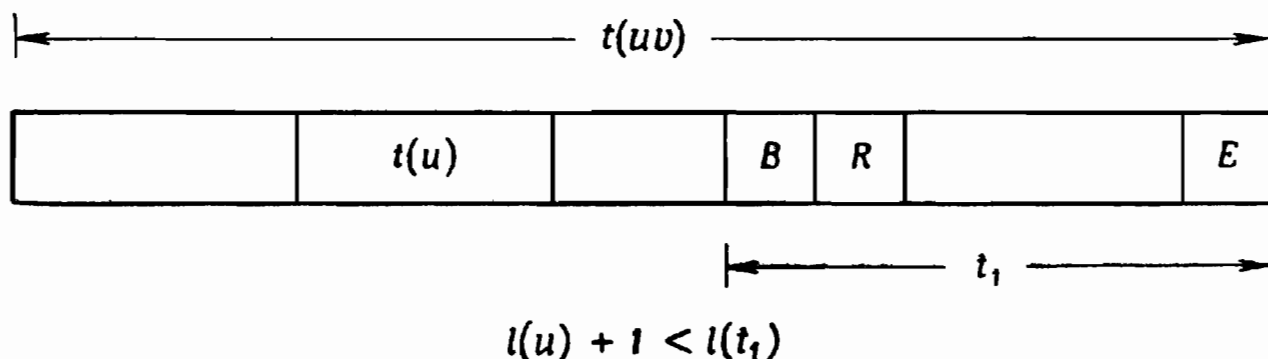
Доказательство. Выберем последовательность $v \in Z$, $l(v) = i$, для которой выполняется (а). Пусть d_1 — число, такое, что больше чем $1/5$ квадратов $t(uv)$ посещались машиной M более чем d_1 раз. Тогда общее число шагов работы машины превышает $d_1 \cdot (1/5)l(t(uv)) \geq (1/5)d_1ci$. Но, поскольку M работает

в реальное время, число шагов ее работы над входом uv в точности равно $l(u) + l(v) \leq 2i$. Таким образом, $\frac{1}{5}d_1ci \leq 2i$ и $d_1 \leq 10/c$. Число $d = [10/c + 1]$ удовлетворяет (б).

4. КВАДРАТЫ УЗКОГО МЕСТА

Доказательство нашей главной теоремы основывается на том, что при подаче определенной входной последовательности на рабочей ленте машины образуются квадраты узкого места, через которые информация не может протекать в достаточном количестве. Понятие узкого места можно уточнить следующим образом:

Определение 3. Пусть $u \in A$, $v \in Z$. Квадрат $B \in t(uv)$ называется *квадратом узкого места* рабочей зоны $t(uv)$, если



Р и с. 1.

1) под действием входа uv машина проходит через B не более чем d раз (где d выбрано по лемме 3), 2) B лежит вне рабочей зоны $t(u)$, 3) длина той части $t(uv)$, которую отделяет B и которая не содержит $t(u)$, превышает $l(u) + 1$.

Лемма 1. Для любого $u \in A$ существует $v \in Z$, такое, что зона $t(uv)$ имеет квадрат узкого места.

Доказательство. Пусть i — натуральное число, такое, что $5l(u) + 5 < ci$, и, кроме того, $l(u) \leq i$. По лемме 3 существует последовательность $v \in Z$, такая, что $ci \leq l(t(uv))$ и менее чем $1/5$ часть квадратов $t(uv)$ посещалась машиной более чем d раз. Получаем $l(t(u)) \leq l(u) + 1 < ci/5 \leq l(t(uv))/5$. Разделив $t(uv)$ на пять равных частей (если $l(t(uv))$ не делится на 5, рассуждение изменяется очевидным образом), мы видим, что либо на левом, либо на правом конце $t(uv)$ есть отрезок длины $(2/5)l(t(uv))$, который не содержит квадратов $t(u)$. В этом отрезке рассмотрим $1/5$ часть $t(uv)$, которая не примыкает к концу этой зоны. Поскольку менее чем $1/5$ квадратов $t(uv)$ посещались машиной более чем d раз, существует квадрат B , лежащий

в выбранной нами $1/5$ части зоны, в котором машина побывала самое большее d раз. Наконец, между B и концом $t(uv)$ лежит по крайней мере $(1/5)l(t(uv)) \geqslant ci/5 > l(u) + 1$ квадратов. Таким образом, B — квадрат узкого места.

5. ДОКАЗАТЕЛЬСТВО ГЛАВНОЙ ТЕОРЕМЫ

Пусть $u \in A$, $v \in Z$, такие, что на зоне $t(uv)$ есть квадрат узкого места B . Для определенности предположим, что B справа от зоны $t(u)$. Когда на вход подается uv , машина в некоторый момент времени приходит в первый раз на крайний правый квадрат E зоны $t(uv)$ (рис. 1). Пусть $w \in Z$ — начальный отрезок последовательности v , такой, что подача uw приводит к первому посещению квадрата E . Таким образом, $t(uv)$ и $t(uw)$ имеют одинаковый крайний правый квадрат E и B является квадратом узкого места также и для $t(uw)$.

Обозначим первый квадрат справа от B через R . **Переходом машины M через B** будем называть либо движение M из B в R , либо движение из R в B . **Состояние машины M при переходе** — это состояние, в которое машина переходит, достигая R (в первом случае) или достигая B (во втором). Заметим, что переходом M через B не считается элементарная операция машины, которую машина начинает и заканчивает в B .

Под действием входа uw машина M будет сначала посещать зону $t(u)$, а затем, когда на вход будет подаваться последовательность w , машина передвинется в квадрат E . Пусть $p_1, p_2, \dots, p_r$ — последовательные переходы через B ($r = 1$ не исключается). Переходу p_1 соответствует сдвиг из B в R , переходу p_2 — сдвиг из R в B и т. д. Пусть состояние машины M при переходе p_i будет s_i , $1 \leqslant i \leqslant r$. **След** ¹⁾ в квадрате узкого места B — это набор длины $r + 1$ вида $(e, s_1, s_2, \dots, s_r)$, где $e = 1$, если B справа от $t(u)$, или $e = -1$, если B слева от $t(u)$, а $s_1, s_2, \dots, s_r$ определены выше. Понятия перехода и состояния при переходе очевидным образом переносятся на случай, когда B слева от $t(u)$.

Теперь вспомним, что число r переходов через B не больше чем d . Тогда существует не более N

$$N = 2m + 2m^2 + \dots + 2m^d$$

различных следов в квадратах узкого места, где m — число состояний машины M .

Пусть g — число, такое, что $N < 2^g$. Для любого $u \in A$, $l(u) = g$ пусть $v \in Z$ есть 01-последовательность, такая, что $t(uv)$ имеет квадрат узкого места B_u , и пусть w означает отрезок

¹⁾ В оригинале *scheme*. — Прим. ред.

входной последовательности u , подача которого приводит к первому посещению машиной крайнего правого квадрата E_u зоны $t(uv)$. Должны быть две различные последовательности $u_1, u_2 \in A$, $l(u_1) = l(u_2) = g$, такие, что квадраты узкого места B_{u_1} и B_{u_2} имеют одинаковые следы, например $(1, s_1, s_2, \dots, s_r)$. Заметим, что $e = 1$, т. е. B_{u_i} справа от $t(u_i)$, $i = 1, 2$.

Пусть

$$u_1 w_1 = u_1 \varepsilon_1 \dots \varepsilon_{n_1} \dots \varepsilon_{n_2} \dots \varepsilon_{n_r} \dots \varepsilon_{n_{r+1}},$$

$$u_2 w_2 = u_2 \delta_1 \dots \delta_{m_1} \dots \delta_{m_2} \dots \delta_{m_r} \dots \delta_{m_{r+1}},$$

где $\varepsilon, \delta \in \{0, 1\}$, ε_{n_1} — входной символ в момент посещения машиной M квадраты B_{u_1} при первом переходе через B_{u_1} , ε_{n_2} — входной символ в момент времени, когда M находится в B_{u_1} при втором переходе через B_{u_1} , и т. д. до ε_{n_r} ; аналогично для $\delta_{m_1}, \delta_{m_2}, \dots$ во второй последовательности $u_2 w_2$. После подачи на вход $\varepsilon_{n_{r+1}} (\delta_{m_{r+1}})$ машина впервые посещает крайний правый квадрат $E_{u_1} (E_{u_2})$.

Мы подошли к главной части нашего доказательства. В последовательности $u_1 w_1$ заменим для каждого нечетного $1 \leq i \leq r-2$ отрезок $\varepsilon_{n_{i+1}} \dots \varepsilon_{n_{i+1}-1}$ последовательностью $\delta_{m_{i+1}} \dots \delta_{m_{i+1}-1}$. Далее, заменим $\varepsilon_{n_{r+1}} \dots \varepsilon_{n_{r+1}}$ на $\delta_{m_{r+1}} \dots \delta_{m_{r+1}}$. Назовем получившуюся последовательность $u_1 w'_1$. Заметим, что все изменения были сделаны в части w_1 входной последовательности. Мы получаем, что $u_1 w_1$ и $u_2 w_2$ имеют одинаковые следы при переходах через B_{u_1} и B_{u_2} соответственно, а наши изменения касались только символов, подаваемых в моменты времени между посещениями B_{u_1} (при этом машина была справа от B_{u_1}) или подаваемых после последнего посещения B_{u_1} . Легко установить с помощью индукции по i , $1 \leq i \leq r+1$, что $u_1 w'_1$ имеет тот же след $(1, s_1, \dots, s_r)$ и что для каждого входного символа ε_{n_j} , j нечетное и $2 \leq j \leq r+1$, часть ленты справа от B_{u_1} имеет тот же вид, что и часть ленты $t(u_2 w_2)$ справа от B_{u_2} при подаче на вход δ_{m_j} , кроме того, состояния M в моменты подачи соответствующих входов совпадают. Зоны $t(u_1 w'_1)$ и $t(u_2 w_2)$ имеют квадраты B_{u_1} и B_{u_2} соответственно со следующими свойствами. Зона $t(u_i)$ лежит вся слева от B_{u_i} , $i = 1, 2$. Части зон $t(u_1 w'_1)$ и $t(u_2 w_2)$, лежащие справа от соответственно B_{u_1} и B_{u_2} , имеют длину, строго превосходящую $l(u_1) = l(u_2) = g$. Как уже было сказано, после подачи $u_1 w'_1$ и $u_2 w_2$ машина M находится в крайних квадратах E_{u_1} и E_{u_2} соответствующих зон, части ленты от B_{u_1} до E_{u_1} и от B_{u_2} до E_{u_2} имеют один и тот же

вид, состояния машины в обоих случаях совпадают. Предположим теперь, что как за $u_1 w'_1$, так и за $u_2 w_2$ следует входная последовательность au_1^* . Поскольку $u_1 \neq u_2$, имеем

$$u_1 w'_1 au_1^* \in T_2, \quad u_2 w_2 au_1^* \notin T_2,$$

но $l(au_1^*) = g + 1$ меньше, чем расстояние между E_{u_i} и B_{u_i} , $i = 1, 2$. Поскольку M работает в реальное время и выполняет не более одного передвижения на один входной символ, после подачи на вход au_1^* она будет оставаться справа от B_{u_i} . Таким образом, машина M в обоих случаях начинает работу в одном и том же состоянии и при одинаковых записях на ленте. Следовательно, после подачи $u_1 w'_1 au_1^*$ и $u_2 w_2 au_1^*$ машина также будет в обоих случаях находиться в одном и том же состоянии, а значит, не сможет допустить одну последовательность и отвергнуть другую. Мы пришли к противоречию.

Анализ приведенного выше доказательства показывает, что мы можем получить из него точную информацию относительно того, где данная одноленточная машина не сможет правильно решить, содержится ли последовательность x в T_2 . Мы формулируем результат без детального доказательства, поскольку оно уже содержится в предыдущем изложении.

Теорема 2. Пусть M имеет t состояний и n символов в рабочем алфавите. Пусть

$$c = \frac{1}{2} \frac{\ln 2}{\ln n}, \quad d = \left\lceil \frac{10}{c} + 1 \right\rceil \quad \text{и} \quad N = 2t + 2t^2 + \dots + 2t^d.$$

Пусть g — наименьшее натуральное число, такое, что $N < 2^g$, и пусть i — наименьшее число, такое, что $5g < ci$ и $it \leq n^i$. Тогда существует последовательность $x \in \Sigma^*$, такая, что $l(x) \leq 2g + i + 1$ и M допускает x , если $x \notin T_2$, или отвергает x , если $x \in T_2$.

6. ОБЩЕЕ ЗАМЕЧАНИЕ ПО ПОВОДУ ДОКАЗАТЕЛЬСТВ НЕВОЗМОЖНОСТИ

Нам бы хотелось коротко сравнить результат теоремы 1 и ее доказательство с другими результатами, касающимися невозможности вычислений на различных математических машинах. Имеющиеся в литературе доказательства невозможности распадутся на два класса. В некоторых ситуациях, когда мы хотим для двух классов A и B математических машин или вычислительных процедур показать, что существует функция, вычисли-

мая с помощью процедуры из B , но не вычислимая никакой процедурой из A , мы можем использовать диагональный метод. Класс B достаточно богат, чтобы содержать некоторую процедуру, в каком-то смысле универсальную относительно A , и эта процедура используется для диагонализации всех процедур в A . Доказательство того, что существует общерекурсивная функция, которая не примитивно рекурсивна, принадлежит именно к такому типу. Класс общерекурсивных функций содержит функцию, которая нумерует все примитивно рекурсивные функции, и эта функция используется в диагональном методе.

Второй метод, главным образом применяемый к математическим машинам, состоит в том, чтобы показать, что машины из класса A не могут хранить информацию необходимую для выполнения определенной задачи. Например, рассмотрим множество $T = \{0^n 10^n \mid n = 1, 2, \dots\}$, где 0^n означает последовательность из n символов 0. Чтобы показать, что T не определимо никаким конечным автоматом, мы замечаем, что, как только n превышает число m состояний M , автомат не может запомнить, сколько нулей было в последовательности 0^n , а следовательно, не всегда может решить, выполняется ли в последовательности $0^n 10^k$ равенство $n = k$.

Теорема 1 не прибегает ни к одному из этих методов. Двухленточные машины Тьюринга, работающие в реальное время, не обладают достаточными возможностями для диагонализации множества всех одноленточных машин Тьюринга, работающих в реальное время. При этом объем хранимой информации на одной ленте такой же, как и на двух. Таким образом, не применим ни один из методов.

Когда класс алгоритмов попадает в эту промежуточную область, где алгоритмы недостаточно сильны для использования диагонализации, но не достаточно слабы для доказательств с помощью оценки информационной емкости, доказательства неосуществимости определенных вычислений довольно трудны. Нам пришлось прибегнуть к тонкому анализу рассматриваемых вычислительных процедур; трудности заключались в обзоре всех возможностей. Таким образом, имеется потребность в новой технике для решения этого класса задач. Нужно заметить, что многие из наиболее интересных вопросов о неосуществимости вычислений попадают в эту промежуточную область.

7. МНОЖЕСТВО, РАСПОЗНАВАЕМОЕ НА ОДНОЛЕНТОЧНОЙ МАШИНЕ

Когда мы рассматриваем множество T_2 , у нас возникает интуитивное чувство, что оно не распознаваемо в реальное время

на одноленточной машине. В самом деле, рассматривая способ, которым вход uv , $u \in A$, $v \in Z$, кодировался двуленточной машиной (доказательство леммы 1), мы видим, что на одной ленте машине придется закодировать u и v в том виде, в котором они поступают на вход. В конце кодировки v машина M находится далеко от $t(u)$; если подать теперь αu^* на вход, M не сможет сравнить его с u . Эти рассуждения весьма далеки от доказательства, так как они относятся только к наиболее очевидному способу кодирования, а в доказательстве невозможности мы должны принимать во внимание все мыслимые способы кодирования.

Ниже приведен пример множества T_1 , которое очень похоже на T_2 и к которому неприменимы описанные интуитивные рассуждения. Оказывается, с помощью более сложного кодирования T_1 распознаваемо на одноленточной машине. Пусть $\Sigma = \{a, 0, \alpha, \beta\}$, и пусть

$$T_1 = \{a^n 0^m \alpha a^n \mid n, m = 1, 2, \dots\} \cup \{a^n 0^m \beta 0^m \mid n, m = 1, 2, \dots\}.$$

Заметим, что T_1 имеет то же строение, что и T_2 , только мы не употребляем b и 1 .

Теорема 3. *Множество T_1 распознаваемо в реальное время на одноленточной машине.*

Доказательство. Мы опишем работу машины M , не вдаваясь в подробности ее устройства.

Когда поступает последовательность a^n , M печатает строку x символов a . Но машина будет печатать символ a и сдвигаться вправо лишь для каждого второго входного символа. Так, в конце подачи a^n на рабочей ленте будет последовательность x из $n/2$ символов a , и M стоит в правом конце x . Когда поступает первый входной 0 , M печатает 0 в правом конце x . Во время подачи 0^m на вход M будет сдвигать всю последовательность x влево от квадрата, в котором записан 0 . Машина будет делать это, стирая a в правом конце x , проходя в левый конец x , записывая там a и возвращаясь к правому концу x . Все это делается со скоростью одна элементарная операция на один входной символ.

Теперь мы различаем два случая. Если за символом α на вход поступает a^k , машина должна проверить, равна ли длина последовательности x (быть может, смещенной от своего первоначального положения на ленте) числу $k/2$ (или $(k/2) - 1$, если машина находится внутри x в процессе перехода справа налево), т. е. машина проверяет равенство $k = n$, а значит, и условие $a^n 0^m \alpha a^k \in T_1$. Обозначим квадрат, на котором стоит машина M внутри x , через E . Когда на входе α , M отмечает E и сдвигается вправо по одному квадрату на входной символ, пока не

доберется до правого конца. Ясно, что M перейдет в левый конец x , а затем опять в квадрат E . Получаем, что $n = k$ в том и только в том случае, когда после подачи k входных нулей M оказывается в E . Если на вход поступает β и затем следует 0^k , то M должна проверить, равно ли k числу m ранее поданных на вход нулей. Машина делает это, сдвигая x вправо обратно к квадрату, в котором записан 0. Сдвиг вправо производится обращением процедуры, ранее использованной для сдвига влево. x возвратится в первоначальное положение при подаче k входных 0 в том и только в том случае, когда $k = m$, т. е. $a^n 0^m \beta 0^k \in T_1$.

Во всех других случаях, когда вся входная последовательность не имеет вида $a^n 0^m \alpha a^k$ или $a^n 0^m \beta 0^k$, можно легко устроить дело так, чтобы машина отвергала эту последовательность. Таким образом, M воспринимает те и только те входные последовательности, которые образуют T_1 .

ЛИТЕРАТУРА

1. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, 117, № 5 (1965). (Русский перевод: Кибернетический сборник (новая серия), вып 4, М., 1967.)
2. Yamada H., Real-time computation and recursive functions not real-time computable, *IRE Trans. El. Computers*, EC-11 (1962), 753—760. (См. стр. 139—155 настоящего сборника.)

ЯЗЫКИ, ОПРЕДЕЛИМЫЕ В РЕАЛЬНОЕ ВРЕМЯ¹⁾

А. Розенберг

1. ВВЕДЕНИЕ

Основная проблема теории искусственных языков состоит в нахождении эффективного алгоритма синтаксического анализа. Если автоматическое устройство на каждом шаге воспринимает новый входной символ и решает вопрос о принадлежности к языку немедленно после считывания последнего входного символа, то такой автомат в каком-то смысле распознает язык за минимально возможное время. Автоматы, которые действуют таким образом, называются автоматами, работающими в реальном масштабе времени, а класс языков, задаваемых такими автоматами, называется классом языков, определяемых в реальное время. В работе исследуются свойства класса автоматов, работающих в реальном масштабе времени, и класса языков, определяемых этими автоматами.

В разд. 2 дается определение и описание свойств рассматриваемой модели.

В разд. 3 доказывается замкнутость относительно операций класса $\mathbf{R}$ языков, определяемых в реальное время. В частности, показывается, что $\mathbf{R}$ является булевой алгеброй, что $\mathbf{R}$ замкнут относительно минимизации, относительно инверсии конечного преобразования в реальном времени и относительно умножения справа на регулярное множество. С использованием замкнутости $\mathbf{R}$ относительно объединения строится пример существенно неоднозначного бесконтекстного языка, определяемого в реальное время.

В разд. 4 содержатся результаты, касающиеся незамкнутости класса $\mathbf{R}$. Доказывается, что $\mathbf{R}$ незамкнут относительно сцепления даже с регулярными множествами, незамкнут также относительно итерации²⁾ и относительно обращения. Рассматриваются также некоторые менее важные операции, а именно отображе-

¹⁾ Rosenberg A. L., Real-time definable languages, *Journal ACM*, 14, № 4 (Oct. 1967), 645—662.

²⁾ Если A — язык, то A^k обозначает k -кратное сцепление A с самим собой, а A^0 — язык, состоящий из одного пустого слова. Тогда итерация A

есть $A^* = \bigcup_{k \geq 0} A^k$.

ние, осуществляемое при помощи последовательностной машины, операции взятия производного и частного и операция максимизации. Как следствие этих результатов получается пример языка, который определим в «дважды реальное время», но не принадлежит классу R . В заключение показывается, что существует детерминированный бесконтекстный язык (Гинзбург и Грейбах [2]), не принадлежащий классу R , и тем самым устанавливается место R в лингвистической иерархии. Доказательства отрицательных результатов опираются только на подсчет количества доступной для автомата информации как функции времени. Это соображение позволяет перенести наши результаты на более широкий класс машин, включающий большинство устройств, описанных в литературе при исследовании работы в реальном масштабе времени. Такой подход выдвигает некоторые проблемы, касающиеся этих новых классов автоматов.

В разд. 5 отмечаются некоторые проблемы, относящиеся к определимости в реальное время, и доказывается их алгоритмическая неразрешимость. В частности, заметим, что неразрешима проблема, является или нет данный язык определяемым в реальное время.

2. ИСХОДНАЯ МОДЕЛЬ

2.1. Предполагается знакомство с понятиями алфавита, слова в алфавите (цепочки) и множества слов (языка), так же как и с различными операциями над словами и языками.

Обозначим через $L(t)$ длину слова t . Для конечного множества A обозначим через $\ast(A)$ число элементов множества A . Если через $A \times B$ обозначить прямое произведение множеств A и B , то $[A]^h$ определяется следующими рекурсивными соотношениями:

$$\begin{aligned} [A]^1 &= A, \\ [A]^{i+} &= A \times [A]^i. \end{aligned}$$

2.2. Исследуемая модель представляет собой многоленточную машину Тьюринга со входом, рассмотренную у Хартманиса и Стирнза [5] и у Рабина [8].

При данном n n -ленточная машина Тьюринга имеет управляющее устройство с конечным числом состояний и n бесконечных в обе стороны лент, на каждой из которых расположена головка, способная считывать и записывать. До начала работы машина находится в выделенном начальном состоянии, а головки стоят на произвольных клетках первоначально пустых рабочих лент (мы будем обозначать пустой символ через $\perp$). Множество

состояний машины разделяется на две группы: «голосующие» состояния и «автономные» состояния.

Находясь в голосующем состоянии, машина воспринимает очередной входной символ. На основе текущего состояния, входного символа и символов, считываемых с рабочих лент, машина изменяет свое состояние, и каждая головка независимо от других записывает на своей рабочей ленте не более одного символа из вспомогательного алфавита и сдвигается не более чем на одну клетку вправо или влево. Начальное состояние обязательно должно быть голосующим.

Находясь в автономном состоянии, машина не воспринимает входного символа; но она выполняет очередной шаг работы так, как описано выше, в зависимости только от текущего состояния и от символов, считываемых с рабочих лент.

Голосующие состояния в свою очередь подразделяются на допускающие и отвергающие состояния. Входная последовательность допускается (соответственно отвергается) n -ленточной машиной Тьюринга, если, исходя из начального состояния с пустыми рабочими лентами, при данной последовательности, подаваемой на вход символ за символом, машина совершает последовательность элементарных шагов, описанных выше, и заканчивает работу в допускающем состоянии (соответственно не делает этого).

Многоленточная машина Тьюринга со входом работает в реальном масштабе времени, если все ее состояния голосующие. Хорошо известно, что ограничение «реального времени» очень сильно суживает возможности машины Тьюринга.

Дадим формальное определение

Определение 1. n -ленточной машиной Тьюринга со входом (короче, просто машиной) называется восьмерка $(K_p, K_a, \Sigma, W, M, S, s_0, F)$, где

(1) K_p и K_a — непересекающиеся конечные множества (соответственно *голосующие* и *автономные* состояния);

(2) Σ — алфавит входных символов; W — рабочий или вспомогательный алфавит, содержащий I ;

(3) M — функция перехода из состояния в состояние, отображающая $(K_p \times \Sigma \times [W]^n) \cup (K_a \times [W]^n)$ в $K_p \cup K_a$;

(4) S — функция действия, отображающая $(K_p \times \Sigma \times [W]^n) \cup (K_a \times [W]^n)$ в $\{R, L, N\}^n \times [W]^n$;

(5) s_0 — начальное состояние, элемент множества K_p ;

(6) F — множество допускающих состояний, подмножество множества K_p .

Если на Σ и W не наложено никаких дополнительных требований, мы часто будем обозначать машину просто (K_p, K_a, M, S, s_0, F) .

Конфигурация машины $(K_p, K_a, \Sigma, W, M, S, s_0, F)$ есть упорядоченный набор из $(n+2)$ элементов $C = (q, t, x_1 t y_1, \dots, x_n t y_n)$, где q — состояние, t — слово в алфавите Σ , x_i и y_i — непустые цепочки из W^* , а t — специальный маркер (не из W), указывающий положение головок на рабочих лентах. Если q — элемент K_p , а q'' — элемент K_a , пишем

$$(q, at, x_1 u_1 t v_1 y_1, \dots, x_n u_n t v_n y_n) \rightarrow (q', t, x'_1 t y'_1, \dots, x'_n t y'_n)$$

или

$$(q'', t, x_1 u_1 t v_1 y_1, \dots, x_n u_n t v_n y_n) \rightarrow (q', t, x'_1 t y'_1, \dots, x'_n t y'_n),$$

где $a \in \Sigma$, $t \in \Sigma^*$, u_i и v_i из W , x_i и y_i из W^* , а x'_i и y'_i — непустые цепочки из W^* , если выполнены следующие условия:

- (1) $q' = M(q, a, v_1, \dots, v_n)$ (соотв. $q' = M(q'', v_1, \dots, v_n)$);
- (2) если $S(q, a, v_1, \dots, v_n)$ (соотв. $S(q'', v_1, \dots, v_n)$) равно $(D_1, D_2, \dots, D_n, w_1, \dots, w_n)$, где каждый символ D_i принадлежит множеству $\{R, L, N\}$, а каждый символ w_i принадлежит W , то для $i = 1, 2, \dots, n$ слово $x'_i t y'_i$ графически равно $x_i u_i t w_i y_i$, $x_i t u_i w_i y_i$ или $x_i u_i w_i t y_i$ в соответствии с тем, равно ли D_i символу N , L или R ¹⁾.

Будем писать

$$C = (q, t, x_1 t y_1, \dots, x_n t y_n) \Rightarrow C' = (q', t', x'_1 t y'_1, \dots, x'_n t y'_n),$$

если либо $C = C'$, либо существует последовательность конфигураций $C_1, \dots, C_k$, в которой $C_1 = C$, $C_k = C'$ и $C_i \rightarrow C_{i+1}$ для каждого i .

Вычислением цепочки t в алфавите Σ на машине Тьюринга со входом называется последовательность конфигураций $(s_0, t, Im I, \dots, Im I) = C_0, C_1, \dots, C_j, \dots$, где $C_i \rightarrow C_{i+1}$ для каждого i . Вычисление называется **полным**, если для некоторого k $C_k = (q, \lambda, x_1 t y_1, \dots, x_n t y_n)$, где λ — пустая цепочка, а q — голосующее состояние из K_p . (Легко проверить, что вычисление полно тогда и только тогда, когда машина останавливается, и что состояние q , если оно существует, определяется единственным образом.) Машина **допускает** цепочку t , если существует полное вычисление t , которое заканчивается допускающим состоянием, т. е. если

$$(s_0, t, Im I, \dots, Im I) \Rightarrow (q, \lambda, x_1 t y_1, \dots, x_n t y_n),$$

где q принадлежит множеству F . Машина **отвергает** t , если она не допускает t .

¹⁾ Если x_i или y_i пуста, заменим ее символом I , чтобы сохранить наше определение конфигурации.

Определение 2. Пусть $\mathcal{J}$ — вычислимая монотонно возрастающая функция натурального аргумента с натуральными значениями. Машина Тьюринга со входом *работает с временной функцией* $\mathcal{J}(p)$, если для любого входного слова t длины p существует полное вычисление t на данной машине длиной не более чем $\mathcal{J}(p) + 1$.

Наибольший интерес для нас представляют те машины, которые работают с временем $\mathcal{J}(p) = p$.

Определение 3. Многоленточная, а именно n -ленточная, машина Тьюринга со входом *есть машина, работающая в реальном масштабе времени* (сокращенно МРВ или n -МРВ), если она имеет вид $(K_p, \emptyset, M, S, s_0, F)$. Для краткости мы часто будем обозначать подобные машины просто (K, M, S, s_0, F) .

Для МРВ понятие допускаемой или отвергаемой цепочки может быть усилено следующим образом: цепочка $t = a_1 \dots a_m$ ($m \geq 0$) из Σ^* допускается (отвергается) n -МРВ T , если вычисление t на машине T есть последовательность $C_0, \dots, C_m$ и состояние машины T в конфигурации C_m принадлежит множеству F (соответственно $K \setminus F$).

Обозначим через $A(T)$ множество цепочек — язык, определяемый (или допускаемый) машиной T . Класс языков, допускаемых всевозможными n -МРВ, обозначается R и называется *классом языков, определимых в реальное время* (сокращенно ЯРВ).

Замечание. Число клеток рабочих лент, которые n -МРВ может пройти при обработке входа t , очевидно, не превышает $nL(t)$. Отсюда вытекает, что любой язык из R определяется детерминированным линейно-ограниченным автоматом (Курода [6]) и что R есть подкласс класса контекстных языков.

Как будет показано в разд. 4.8, этот подкласс собственный.

3. СВОЙСТВА ЗАМКНУТОСТИ ОТНОСИТЕЛЬНО ОПЕРАЦИЙ

В данном разделе доказывается замкнутость класса R относительно некоторых операций.

3.1. Булевы операции

Теорема 1. *Класс языков, определимых в реальное время, замкнут относительно дополнения, пересечения и объединения. Иначе говоря, класс R образует булеву алгебру.*

Доказательство. Пусть X и Y — языки, определяемые соответственно n -МРВ $T_X = (K_X, M_X, S_X, s_0, F_X)$ и m -МРВ $T_Y = (K_Y, M_Y, S_Y, t_0, F_Y)$. Тогда:

(а) $\bar{X}$, дополнение языка X , определяется n -МРВ

$$\bar{T}_X = (K_X, M_X, S_X, s_0, K_X \setminus F_X).$$

В самом деле, $\bar{T}_X$ работает точно так же, как T_X , только вместо ответа «да» дает ответ «нет» и наоборот. Очевидно,

$$A(\bar{T}_X) = \overline{A(T_X)} = \bar{X}.$$

(б) В предположении, что T_X и T_Y работают в одном и том же входном алфавите, пересечение языков X и Y допускается $(n+m)$ -ленточной машиной $T_V = (K_V, M_V, S_V, v_0, F_V)$, у которой

$$(1) K_V = K_X \times K_Y;$$

$$(2) \text{ для } \langle s, t \rangle \text{ из } K_X \times K_Y, w_1, \dots, w_n \text{ из } W_X \text{ и } w'_1, \dots, w'_m \text{ из } W_Y \\ M_V(\langle s, t \rangle, \sigma, w_1, \dots, w_n, w'_1, \dots, w'_m) = \{M_X(s, \sigma, w_1, \dots, w_n)\} \times \\ \times \{M_Y(t, \sigma, w'_1, \dots, w'_m)\};$$

$$(3) \text{ если } S_X(s, \sigma, w_1, \dots, w_n) = \langle D_1, \dots, D_n, \beta_1, \dots, \beta_n \rangle \text{ и } \\ S_Y(t, \sigma, w'_1, \dots, w'_m) = \langle D'_1, \dots, D'_m, \delta_1, \dots, \delta_m \rangle, \text{ то}$$

$$S_V(\langle s, t \rangle, \sigma, w_1, \dots, w_n, w'_1, \dots, w'_m) = \\ = \langle D_1, \dots, D_n, D'_1, \dots, D'_m, \beta_1, \dots, \beta_n, \delta_1, \dots, \delta_m \rangle;$$

$$(4) v_0 = \langle s_0, t_0 \rangle;$$

$$(5) F_V = F_X \times F_Y.$$

Машина T_V имитирует работу T_X на лентах $1, 2, \dots, n$, а работу T_Y — на лентах $(n+1), \dots, (n+m)$. В результате T_V допускает входное слово тогда и только тогда, когда его допускают и T_X и T_Y ; следовательно,

$$A(T_V) = A(T_X) \cap A(T_Y) = X \cap Y.$$

(в) Замкнутость R относительно объединения следует из п. (а) и (б) и из законов де Моргана. Ч. т. д.

Очевидно, множества $A = \{a^i b^j c^k : i, j \geq 1\}$ и $B = \{a^i b^j c^k : i, j \geq 1\}$ оба определимы в реальное время. Поскольку $A \cap B$ не является бесконтекстным языком, а $A \cup B$ существенно неоднозначен (см. Гинзбург и Уллиан [4]), теорема 1 дает следующие два следствия:

Следствие 1.1. *Существуют языки, определяемые в реальное время и не являющиеся бесконтекстными.*

Следствие 1.2. *Существуют существенно неоднозначные бесконтекстные языки, определяемые в реальное время.*

Рабин [8] доказал, что если T_1 и T_2 — одноленточные МРВ, то может случиться, что язык $A(T_1) \cup A(T_2)$ (а значит, и

$A(T_1) \cap A(T_2)$) определим двуленточной МРВ, но не определим одноленточной МРВ. Это дает основание предполагать, что конструкция (б) из теоремы 1 дает наилучшее решение вопроса о количестве рабочих лент.

3.2. Умножение справа на регулярное множество. В этом разделе доказывается, что если множество A определимо в реальное время, то для любого регулярного множества B множество AB определимо в реальное время. Как будет показано в разд. 4.1, множество BA не обязательно будет определимым в реальное время; это первый «асимметричный» результат, касающийся R .

Определение 4. Конечным автоматом во входном алфавите Σ называется четверка $T = (K, M, s_0, F)$, где

- (1) K — конечное множество состояний;
- (2) M — функция перехода, отображающая $K \times \Sigma$ в K ;
- (3) $s_0 \in K$ — начальное состояние;
- (4) $F \subseteq K$ — множество допускающих состояний.

Любой конечный автомат, таким образом, представляет собой 0-МРВ.

Язык, определяемый конечным автоматом, называется регулярным множеством. Таким образом, любое регулярное множество определимо в реальное время.

Теорема 2. Если A — язык, определимый в реальное время, а R — регулярное множество, то язык AR определим в реальное время.

Доказательство. Пусть A задается при помощи n -МРВ

$$T_A = (K_A, M_A, S_A, s_0, F_A),$$

а R задается конечным автоматом

$$T_R = (K_R, M_R, t_0, F_R).$$

Тогда n -МРВ

$$T_{AR} = (K_{AR}, M_{AR}, S_{AR}, u_0, F_{AR}),$$

определенная ниже, распознает множество AR .

(1) $K_{AR} = K_A \times 2^{K_R}$, где 2^{K_R} означает множество всех подмножеств множества K_R .

Пусть $u = \langle s, X \rangle$, где $s \in K_A$ и $X \subseteq K_R$. Для σ из Σ и $w_1, \dots, w_n$ из W

$$(2) M_{AR}(u, \sigma, w_1, \dots, w_n) =$$

$$(I) \{M_A(s, \sigma, w_1, \dots, w_n)\} \times \left(\{t_0\} \cup \bigcup_{t \in X} \{M_R(t, \sigma)\} \right), \text{ если}$$

$M_A(s, \sigma, w_1, \dots, w_n)$ принадлежит F_A ;

$$(II) \{M_A(s, \sigma, w_1, \dots, w_n)\} \times \left(\bigcup_{t \in X} \{M_R(t, \sigma)\} \right) \text{ в противном случае.}$$

$$(3) S_{AR}(u, \sigma, w_1, \dots, w_n) = S_A(s, \sigma, w_1, \dots, w_n).$$

$$(4) u_0 = \begin{cases} \langle s_0, \{t_0\} \rangle, & \text{если } s_0 \text{ принадлежит } F_A, \\ \langle s_0, \emptyset \rangle & \text{в противном случае.} \end{cases}$$

$$(5) F_{AR} = \{u \in K_{AR} : u = \langle s, X \rangle \text{ и } X \cap F_R \neq \emptyset\}.$$

Доказательство того, что $A(T_{AR}) = A(T_A) A(T_R) = AR$, опускается ввиду очевидности. Ч. т. д.

3.3. Инверсия преобразования в реальном масштабе времени. В разделе показано, что прообраз ЯРВ при отображении, осуществляемом конечным преобразователем, работающим в реальном масштабе времени, также будет определен в реальное время. Ниже (разд. 4) будет показано, что прямой образ языка из R при отображении, осуществляемом последовательностной машиной (вырожденный случай преобразователя, работающего в реальном масштабе времени), не обязательно будет определен в реальное время.

Многоленточный преобразователь, работающий в реальное время (сокращенно n -ПРВ), представляет собой по существу n -МРВ с выходом. Говоря более формально, n -ПРВ есть восьмерка

$$U = (Q, \Sigma, V, \Delta, N, S, P, q_0),$$

где

(1) Q — множество состояний U ;

(2) Σ — входной алфавит U ;

V — рабочий (вспомогательный) алфавит U ;

Δ — выходной алфавит U ;

(3) N — функция состояния, отображающая $Q \times \Sigma \times [V]^n$ в Q ;

(4) S — функция действия, отображающая $Q \times \Sigma \times [V]^n$ в $\{R, L, N\}^n \times [V]^n$;

(5) P — функция выхода, отображающая $Q \times \Sigma \times [V]^n$ в Δ^* ;

(6) $q_0 \in Q$ — начальное состояние.

В общих чертах n -ПРВ работает так же, как и n -МРВ, но имеются два основных отличия. Во-первых, n -ПРВ не имеет

выделенных допускающих состояний. Во-вторых (что важнее), на каждом элементарном шаге n -ПРВ выдает выходное слово — слово в выходном алфавите Δ , возможно, пустое.

Пусть $U = (Q, \Sigma, V, \Delta, N, S, P, q_0)$ есть n -ПРВ. Выходная функция P может быть очевидным образом продолжена на множество всех входных последовательностей; а именно если q из Q , σ из Σ , $w_1, \dots, w_n$ из V , то для любой цепочки σx в алфавите Σ

$$P(q, \sigma x, w_1, \dots, w_n) = P(q, \sigma, w_1, \dots, w_n) P(t, x, u_1, \dots, u_n),$$

где $t = N(q, \sigma, w_1, \dots, w_n)$, а $u_1, \dots, u_n$ — символы, читаемые на рабочих лентах машины U после шага, определяемого состоянием q , входным символом σ и рабочими символами $w_1, \dots, w_n$. Для каждой цепочки x в алфавите Σ обозначим через $U(x)$ слово $P(q_0, x, I, \dots, I)$ в алфавите Δ . Затем обозначим через L_U число

$$L_U = \max_{\substack{q \in Q \\ \sigma \in \Sigma \\ w_i \in V}} \{L(P(q, \sigma, w_1, \dots, w_n))\}.$$

Если заданы язык A и n -ПРВ U , прообразом A при отображении U называется множество

$$U^{-1}(A) = \{x: U(x) \text{ принадлежит } A\}.$$

Наше доказательство замкнутости R при отображении, обратном осуществляемому n -ПРВ, основано на том, что, когда ПРВ работает над входом длины k , на каждой рабочей ленте изменяется не более чем k символов.

Предполагая известным понятие временной функции, введенное у Хартманиса и Стирнза [5], мы начинаем доказательство теоремы с установления ряда вспомогательных результатов (см. определение 2).

Теорема 3 (Хартманис и Стирнз [5]). *Если язык A определим за время $L(t) + E(L(t))$, $E(L(t)) \geq 0$, то для любого $k > 0$ язык A определим за время $L(t) + [k \cdot E(L(t))]^1$.*

В частности, имеем

Следствие 3.1. *Если A определим за время $cL(t)$ при некотором $c \geq 1$, то A определим за время $1 + \left[\frac{c-1}{k}\right] L(t)$ для любого $k > 0$.*

¹⁾ Если x — действительное число, то $[x]$ означает наименьшее целое число k , такое, что $k \geq x$.

Значение теоремы 3 состоит в том, что по любой n -ленточной машине Тьюринга при любом целом r можно построить другую n -ленточную машину, которая делает за один элементарный шаг все те операции на рабочих лентах, которые первоначальная машина делала за r шагов между двумя очередными считываниями входных символов.

К сожалению, n -ленточная машина, распознающая множество A за время $2L(t)$, не всегда может быть заменена какой-либо m -ленточной машиной, работающей в реальном масштабе времени. В разд. 4.3 приводится пример языка A , распознаваемого за время $2L(t)$, но не принадлежащего классу R . Однако если такое сжатие r шагов в один шаг достаточно, чтобы ускорить машину Тьюринга до реального масштаба времени, то алгоритм Хартманиса и Стирнза [5, теорема 2] дает эффективный метод этого ускорения. В частности, если n -ленточная машина Тьюринга воспринимает входную последовательность с постоянной скоростью, этот алгоритм дает нам эквивалентную n -МРВ. Покажем теперь, что машина, определяющая $U^{-1}(A)$ для A из R , относится к этому специальному виду.

Теорема 4. Если A есть ЯРВ, а U — преобразователь, работающий в реальном масштабе времени, то $U^{-1}(A)$ также есть ЯРВ.

Доказательство. Пусть ЯРВ A определяется на n -МРВ $T = (K, \Delta, W, M, S, s_0, F)$, и пусть $U = (Q, \Sigma, V, \Delta, N, R, P, q_0)$ m -ПРВ. Отметим, что Δ является входным алфавитом T и выходным алфавитом U . Тогда $(m+n)$ -ленточная машина $T' = (L_p, L_a, \Sigma, W \cup V, M', S', s'_0, F')$ задает $U^{-1}(A)$ за время $(c+1)L(t)$, где $c = L_U$.

(1) $L_p = K \times Q \times \{\lambda\}$, где λ — пустое слово;

(2) L_a пусто (т. е. T' работает в реальном времени), если $c = 0$; в противном случае $L_a = K \times Q \times \bigcup_{i \leq c} \Delta^i$, где Δ^i — множество

всех слов в Δ длины i .

Пусть a — символ из Σ ; $w_1, \dots, w_n$ — из W , а $v_1, \dots, v_m$ — из V ;

(3) для $\langle s, q, \lambda \rangle$ из L_p

$$\begin{aligned} M'(\langle s, q, \lambda \rangle, a, w_1, \dots, w_n, v_1, \dots, v_m) = \\ = \langle s, N(q, a, v_1, \dots, v_m), P(q, a, v_1, \dots, v_m) \rangle \end{aligned}$$

и

$$\begin{aligned} S'(\langle s, q, \lambda \rangle, a, w_1, \dots, w_n, v_1, \dots, v_m) = \\ = \langle \underbrace{N, \dots, N}_{n \text{ раз}}, D_1, \dots, D_m, w_1, \dots, w_n, u_1, \dots, u_m \rangle, \end{aligned}$$

где

$$R(q, a, v_1, \dots, v_m) = \langle D_1, \dots, D_m, u_1, \dots, u_m \rangle;$$

$$(4) \text{ для } \langle s, q, b_1, \dots, b_r \rangle \text{ из } L_a (r \geq 1 \text{ и } b_i - \text{буквы из } \Delta) \\ M'(\langle s, q, b_1 \dots b_r \rangle, w_1, \dots, w_n, v_1, \dots, v_m) = \\ = \langle M(s, b_1, w_1, \dots, w_n), q, x \rangle,$$

где

$$x = b_2 \dots b_r \text{ для } r > 1 \text{ и } x = \lambda \text{ для } r = 1,$$

и

$$S'(\langle s, q, b_1 \dots b_r \rangle, w_1, \dots, w_n, v_1, \dots, v_m) = \\ = \langle D_1, \dots, D_n, \underbrace{N, \dots, N}_{m \text{ раз}}, u_1, \dots, u_n, v_1, \dots, v_m \rangle,$$

где

$$S(s, b_1, w_1, \dots, w_n) = \langle D_1, \dots, D_n, u_1, \dots, u_n \rangle;$$

$$(5) s'_0 = \langle s_0, q_0, \lambda \rangle;$$

$$(6) F' = F \times Q \times \{\lambda\}.$$

Таким образом, T' попеременно имитирует то работу T на лентах $1, 2, \dots, n$, то работу U на лентах $n+1, \dots, n+m$. Из описания T' явствует, что $A(T') = U^{-1}(A(T)) = U^{-1}(A)$. В то же время T' считывает очередной входной символ не более чем через c шагов на рабочих лентах. Прибавляя лишние «холостые» операции, можно добиться того, чтобы T' читала с постоянной скоростью c шагов на входной символ. Тогда, если бы мы могли сжать c шагов машины T' в один, мы могли бы избавиться от автономных состояний и тем самым ускорить T' от времени $(c+1)L(t)$ до реального масштаба времени. Именно это и делает алгоритм Хартманиса и Стирнза. Таким образом, используя вышеописанную конструкцию и этот алгоритм, мы можем эффективно построить $(n+m)$ -МРВ T'' , такую, что

$$A(T'') = A(T') = U^{-1}(A). \text{ Ч. т. д.}$$

Обобщенной последовательностной машиной (ОПМ) называется 0-ПРВ. Имеем как следствие из теоремы 4

Следствие 4.1. Если A есть ЯРВ, а G есть ОПМ, то $G^{-1}(A)$ также есть ЯРВ.

Следствие 4.2. Если некоторый гомоморфный образ¹⁾ языка A определим в реальное время, то и сам язык A определим в реальное время.

¹⁾ Гомоморфизм множества Σ^* в множество Δ^* — это отображение h , такое, что $h(\lambda) = \lambda$, $h(a)$ — цепочка из Δ^* для каждого a из Σ и $h(a_1 \dots a_k) = h(a_1) \dots h(a_k)$ для каждой цепочки $a_1 \dots a_k$ элементов Σ . Для $A \subseteq \Sigma^*$ образом A при гомоморфизме h является множество $h(A) = \{h(x) : x \text{ из } A\}$.

В разд. 4.4 будет показано, что прообраз ЯРВ при отображении, осуществляемом недетерминированной ОПМ, не всегда будет принадлежать R .

3.4. Минимизация. Доказывается замкнутость R относительно операции минимизации.

Мы скажем, что слово x есть собственное начало слова y (пишем $x < y$), если существует непустое слово z , такое, что $y = xz$. Для языка A определим множество $\min A = \{y: y \in A, \text{ и если } x < y, \text{ то } x \notin A\}$.

Теорема 5. *Класс языков, определяемых в реальное время, замкнут относительно минимизации.*

Доказательство. Пусть A — ЯРВ, задаваемый на n -МРВ $T = (K, M, S, s_0, F)$. Тогда $\min A$ задается при помощи n -МРВ $T_{\min} = (K', M', S', s_0, F)$, где

(1) $K' = K \cup \{d\}$, где d — символ, не принадлежащий K ;

Для σ из Σ и $w_1, \dots, w_n$ из W

(2) $M'(s, \sigma, w_1, \dots, w_n) =$

$$= \begin{cases} M(s, \sigma, w_1, \dots, w_n), & \text{если } s \in K \setminus F, \\ d, & \text{если } s \in F, \end{cases}$$

$$M'(d, \sigma, w_1, \dots, w_n) = d;$$

(3) $S'(s, \sigma, w_1, \dots, w_n) = S(s, \sigma, w_1, \dots, w_n)$ для s из K ,

$$S'(d, \sigma, w_1, \dots, w_n) = \{[N]\}^n \times \{\langle w_1, \dots, w_n \rangle\}.$$

Машина $T_{\min}$ подражает действиям T до тех пор, пока T не допустит какое-либо начало входного слова; после этого $T_{\min}$ переходит в «мертвое» состояние d . Ясно, что $A(T_{\min}) = \min A(T)$. Ч. т. д.

Можно определить оператор $\max$, симметричный оператору $\min$. Как показано в разд. 4, класс R не замкнут относительно $\max$, что дает нам еще один несимметричный результат.

3.5. Сохранение числа лент. До сих пор мы тщательно избегали упоминания о минимальном числе лент, необходимом для распознавания какого-либо ЯРВ. Общий вопрос, будет ли $(n+1)$ -МРВ обладать строго большей порождающей способностью, чем n -МРВ, все еще остается открытым. Рабин [8] показал, что 2-МРВ существенно сильнее, чем 1-МРВ. Однако можно утверждать следующее:

Замечание. Обозначим $R(n)$ класс языков, определяемых при помощи n -МРВ. Тогда для всех n

(а) $R(n)$ замкнут относительно дополнения;

(б) $R(n)$ замкнут относительно объединения и пересечения с регулярными множествами;

(в) $R(n)$ замкнут относительно умножения справа на регулярное множество;

(г) $R(n)$ замкнут относительно отображения, обратного к осуществляемому на ОПМ;

(д) $R(n)$ замкнут относительно минимизации.

Замечание. $R(1)$ не замкнут ни относительно объединения и пересечения, ни относительно отображения, обратного к осуществляемому на ПРВ.

Более подробное изучение $R(n)$ является предметом дальнейших исследований.

4. СВОЙСТВА НЕЗАМКНУТОСТИ

Приведем примеры некоторых операций, не сохраняющих определенность в реальное время.

4.1. Сцепление. Доказывается незамкнутость R относительно сцепления даже с регулярными множествами. Доказательство опирается на ряд определений и лемм.

Определение 5. Пусть A — множество слов. Бинарное отношение $E_k(\text{mod } A)$ для слов x и y определяется следующим образом: $x E_k y(\text{mod } A)$, если для всех слов z длины, не превышающей k , слово xz принадлежит A тогда и только тогда, когда слово yz принадлежит A .

Очевидно, $E_k(\text{mod } A)$ является отношением эквивалентности.

Лемма 1 (Хартманис и Стирнз [5]). *Если T есть n -МРВ с d состояниями и w рабочими символами (включая 1), то число классов эквивалентности¹⁾ по $E_k(\text{mod } A(T))$ не может превосходить $d \cdot w^{(2k+1)n}$.*

Ограничимся рассмотрением того случая, когда алфавит Σ имеет по меньшей мере две буквы. Можно предположить, что $\{0, 1\} \subseteq \Sigma$.

Определим язык²⁾

$$P = \{11t1100x\beta t^T: x \in \Sigma^*, t \in \Sigma^* \setminus \Sigma^*11\Sigma^*, \beta \notin \Sigma\}.$$

Лемма 2. *Язык P определим в реальное время.*

В самом деле P определим в реальное время PDS-автоматом (Гинзбург и Грейбах [2]).

Наша цель показать, что Σ^*P не определим в реальное время. Мы выполним это, исследуя скорость роста индекса отношения $E_k(\text{mod } \Sigma^*P)$.

¹⁾ Называемое индексом отношения. — Прим. ред.

²⁾ t^T — инверсия (обращение) слова t . — Прим. ред.

Определение 6. Для всех целых m обозначим через X_m множество всех слов длины m в алфавите $\{0, 1\}$, таких, что ни одно из них не содержит вхождения двух последовательных единиц.

Числа Фибоначчи определяются так: (1) $a_0 = a_1 = 1$; (2) для $n > 1$ $a_n = a_{n-1} + a_{n-2}$. Из теории уравнений в конечных разностях известно, что для достаточно больших n $a_n > (1,6)^n$.

Лемма 3. Для всякого $m > 0$ число слов в X_m (обозначим его через $\ast(m)$) в точности равно $(m+1)$ -му числу Фибоначчи.

Доказательство. Индукция по m .

Базис: $\ast(1) = 2 = a_2$ и $\ast(2) = 3 = a_3$, так как

$$X_1 = \{0, 1\} \quad \text{и} \quad X_2 = \{00, 01, 10\}.$$

Предположение индукции: пусть для $j \leq r$, $\ast(j) = a_{j+1}$. Индукционный шаг: элементы X_{r+1} могут быть получены из элементов X_r следующими путями:

(1) Если t содержится в X_r , то $0t$ содержится в X_{r+1} . Это дает $\ast(r)$ слов для X_{r+1} .

(2) Если t содержится в X_r , $1t$ содержится в X_{r+1} тогда и только тогда, когда первый символ t есть 0.

Из (1) видно, что это дает в точности $\ast(r-1)$ слов для X_{r+1} . Очевидно, X_{r+1} исчерпывается шагами (1) и (2), откуда $\ast(r+1) = \ast(r) + \ast(r-1) = a_{r+1} + a_r = a_{r+2}$. Ч. т. д.

Существует, как легко видеть, в точности $2^{\ast(m)}$ подмножеств X_m . Мы покажем, что каждое ненулевое подмножество определяет класс эквивалентности отношения $E_{m+1}(\text{mod } \Sigma^*P)$, отличный от всех других.

Определение 7. Для каждого слова t_i из X_m образуем слово $\alpha_i = 11t_i1100$ длины $m+6$.

Для каждого ненулевого подмножества $Y_i = \{t_1, \dots, t_r\}$ множества X_m ($1 \leq i \leq 2^{\ast(m)} - 1$) образуем слово $\delta_i = \alpha_1\alpha_2\alpha_3 \dots \alpha_r$ длины $(m+6)r$.

Лемма 4. Если слово t принадлежит $Y_i \subseteq X_m$, то $\delta_i \beta t^T$ принадлежит Σ^*P .

Очевидна из построения δ_i .

Положив $k = m+1$, мы получим лемму 5.

Лемма 5. Для δ_i и Y_i из определения 7 $\delta_i E_k \delta_j (\text{mod } \Sigma^*P)$ только тогда, когда $Y_i = Y_j$.

Доказательство. Положим $Y_i \neq Y_j$. Тогда существует слово t , принадлежащее либо $Y_i \setminus Y_j$, либо $Y_j \setminus Y_i$. Положим,

не теряя общности, что t принадлежит $Y_i \setminus Y_j$. По лемме 4 $\delta_i \beta t^T$ принадлежит Σ^*P . Утверждается, что $\delta_j \beta t^T$ не содержится в Σ^*P . Пусть $\delta_j \beta t^T$ содержится в Σ^*P . Это предполагает по определению P , что δ_j имеет подслово вида $11x1100$, такое, что x не содержит подряд двух вхождений единиц и $x=t$. Однако по построению $\delta_j = 11t_11100 \dots 11t_r1100$, причем $Y_j = \{t_1, \dots, t_r\}$. Поэтому все подслова δ_j вида $11x1100$, где $L(x) = m$ и x не содержит подряд двух вхождений единиц, имеют в точности вид $11t_a1100$, где t_a содержится в Y_j . Значит, если $\delta_j \beta t^T$ содержится в Σ^*P , то t содержится в Y_j , что противоречит предположению (что $t \in Y_i \setminus Y_j$). Мы пришли к выводу, что $\delta_j \beta t^T$ не содержится в Σ^*P , если t не содержится в Y_j . Поскольку $L(t) = m$, мы можем показать, что $\delta_i E_k \delta_j \pmod{\Sigma^*P}$ только тогда, когда $Y_i = Y_j$; ч. т. д.

Так как все δ_i ($1 \leq i \leq 2^{\otimes(m)} - 1$) содержатся в различных классах эквивалентности отношения $E_k \pmod{\Sigma^*P}$, доказана еще

Лемма 6. Число классов эквивалентности отношения $E_{k+1} \pmod{\Sigma^*P}$ не меньше чем $2^{\otimes(k)} - 1$.

Теперь мы в состоянии показать, что Σ^*P не принадлежит $\mathbf{R}$.

Теорема 6. Класс языков, определяемых в реальное время, не замкнут относительно сцепления.

Доказательство. Предположим противное, т. е. что n -МРВ T определяет Σ^*P . По лемме 1 число классов эквивалентности отношения $E_k \pmod{A(T)}$ не может превзойти $e_k = d \cdot \omega^{(2k+1)n}$. Далее, для больших k $\otimes(k) = a_{k+1} > (1,6)^{k+1}$. Поэтому число классов эквивалентности отношения $E_k \pmod{\Sigma^*P}$ удовлетворяет для больших k неравенству

$$2^{(1,5)^k} \leq 2^{\otimes(k-1)} - 1 \leq \text{индекс } E_k \pmod{\Sigma^*P}.$$

Так как $2^{(1,5)^k} > e_k$ для достаточно больших k , T не может определять Σ^*P вопреки предположению; таким образом, Σ^*P не содержится в $\mathbf{R}$. Мы нашли два ЯРВ Σ^* и P , сцепление которых не является ЯРВ, ч. т. д.

Поскольку Σ^* — регулярное множество, доказано

Утверждение 1. Класс языков, определяемых в реальное время, не замкнут относительно сцепления с регулярным множеством.

4.2. Итерация. Используя результаты разд. 4.1, мы покажем, что $\mathbf{R}$ не замкнут относительно операции итерации множеств. Доказательство вытекает из нескольких общих результатов.

Пусть $\mathcal{C}$ — класс языков. Мы назовем $\mathcal{C}$ *приемлемым* (admissible), если

(а) $\mathcal{C}$ замкнут относительно объединения и пересечения с регулярными множествами,

(б) существует язык L в алфавите Σ , принадлежащий классу $\mathcal{C}$ и такой, что если δ символ не из Σ , то $L\{\delta\}$ принадлежит $\mathcal{C}$, в то время как $\Sigma^*L\{\delta\}$ нет.

Теорема 7. Если $\mathcal{C}$ — приемлемый класс языков, то $\mathcal{C}$ не замкнут относительно итерации.

Доказательство. Пусть L — некоторый элемент $\mathcal{C}$, удовлетворяющий пункту (б) определения приемлемости, и пусть δ — символ, не принадлежащий алфавиту Σ языка L . Поскольку Σ^* — регулярное множество, язык $H = L\{\delta\} \cup \Sigma^*$ принадлежит $\mathcal{C}$. Утверждается, что H^* не принадлежит $\mathcal{C}$.

Положим противное, т. е. что H^* содержится в $\mathcal{C}$. Тот факт, что $\Sigma^*\{\delta\}$ — регулярное множество, необходимо влечет, что $J = H^* \cap \Sigma^*\{\delta\}$ содержится в $\mathcal{C}$. Однако легко видеть, что $J = \Sigma^*L\{\delta\}$ и, следовательно, не может содержаться в $\mathcal{C}$. Мы заключаем отсюда, что $\mathcal{C}$ не замкнут относительно итерации. Ч. т. д.

Лемма 7. $\mathbf{R}$ — приемлемый класс языков.

Доказательство. Замкнутость $\mathbf{R}$ относительно объединения и пересечения следует из теоремы 1. Чтобы проверить пункт (б) определения приемлемости, рассмотрим язык P из разд. 4.1. Так как $\mathbf{R}$ замкнут относительно умножения справа на регулярное множество, то $P\{\delta\}$ содержится в $\mathbf{R}$ для всякого символа δ . Далее, доказательство теоремы 6 показывает, что $\Sigma^*P\{\delta\}$ не содержится в $\mathbf{R}$ (последний символ δ не влияет на доказательство). Ч. т. д.

Ввиду приемлемости $\mathbf{R}$ получаем следующую теорему.

Теорема 8. Класс языков, определяемых в реальное время, не замкнут относительно операции итерации.

Замечание. Доказательства теорем 6 и 8 показывают также, что и класс языков, определяемых в реальное время на PDS-автоматах, не замкнут относительно сцепления и итерации.

4.3. Обращение. Здесь доказываемся, что $\mathbf{R}$ не замкнут относительно обращения. Опираясь на это доказательство, мы покажем, что существуют языки, определяемые за время $2L(t)$ (и поэтому по следствию 3.1 за время $(1 + \epsilon)L(t)$), но не определяемые в реальное время.

Язык Q , определяемый ниже, будет ЯРВ, в то время как его обращение не будет ЯРВ. Q есть множество последовательно-

стей вида $t\beta 0011t_1110011t_211\dots 0011t_k11$, где (1) $t, t_1, \dots, t_k$ принадлежат X_m для некоторого m и $\beta \notin \{0, 1\}$ и (2) по крайней мере для одного $i \in \{1, \dots, k\}$, $t_i = t^T$, т. е. обращению начального отрезка t рассматриваемого слова.

Лемма 8. *Язык Q есть ЯРВ.*

Доказательство. Мы дадим набросок действий двухленточной машины Тьюринга (2-МРВ) T , которая определяет Q . Заинтересованный читатель легко восстановит детали.

Работа T происходит следующим образом:

(1) Пока читается начальный отрезок t входа, T копирует t на каждой из своих рабочих лент, убеждаясь, что t не содержит подряд двух вхождений единицы.

(2) Как только поступает t_1 , T сравнивает на ленте 1, равно ли t_1 обращению t ($t_1 = t^T$). Если $L(t_1) \neq L(t)$, T переходит в мертвое состояние; если $t_1 = t^T$, T переходит к шагу (4); если $t_1 \neq t^T$, T переходит к шагу (3).

(3) Как только поступает t_{2i} (соотв. t_{2i+1}), T сравнивает на ленте 2 (соотв. на ленте 1) t_{2i} (соотв. t_{2i+1}) с t^T , одновременно восстанавливая ленту 1 (соотв. ленту 2) так, что головка находится на крайнем правом квадрате. Если $L(t_{2i})$ (соотв. $L(t_{2i+1})$) $\neq L(t)$, T переходит в мертвое состояние; если t_{2i} (соотв. t_{2i+1}) $= t^T$, T переходит к шагу (4); в противном случае T повторяет шаг (3).

(4) Убедившись в том, что некоторое $t_i = t^T$, T проверяет на первой ленте, чтобы все подслова t_j были длины $L(t)$. T переходит в допускающее состояние только тогда, когда она заканчивает воспринимать отрезок t_i11 , причем $L(t_i) = L(t)$.

Расположение головок и лент показано на рис. 1. Ясно, что T определяет Q в реальное время. Ч. т. д.

Теорема 9. *Класс языков, определимых в реальное время, не замкнут относительно обращения.*

Доказательство. Заметим, что Q^T является ¹⁾ подмножеством Σ^*P из разд. 4.1. Доказательство теоремы 6 показывает, что Q^T не есть ЯРВ. Незамкнутость R относительно обращения следует, таким образом, из леммы 8 и доказательства теоремы 6. Ч. т. д.

Лемма 9. *Язык Q^T определим за время $2L(t)$.*

Доказательство. Трехленточная машина Тьюринга, определяющая Q^T , просто копирует вход на одной из своих лент

¹⁾ Q^T — обращение языка Q , состоящее из всех слов, являющихся обращениями слов из Q . — *Прим ред.*

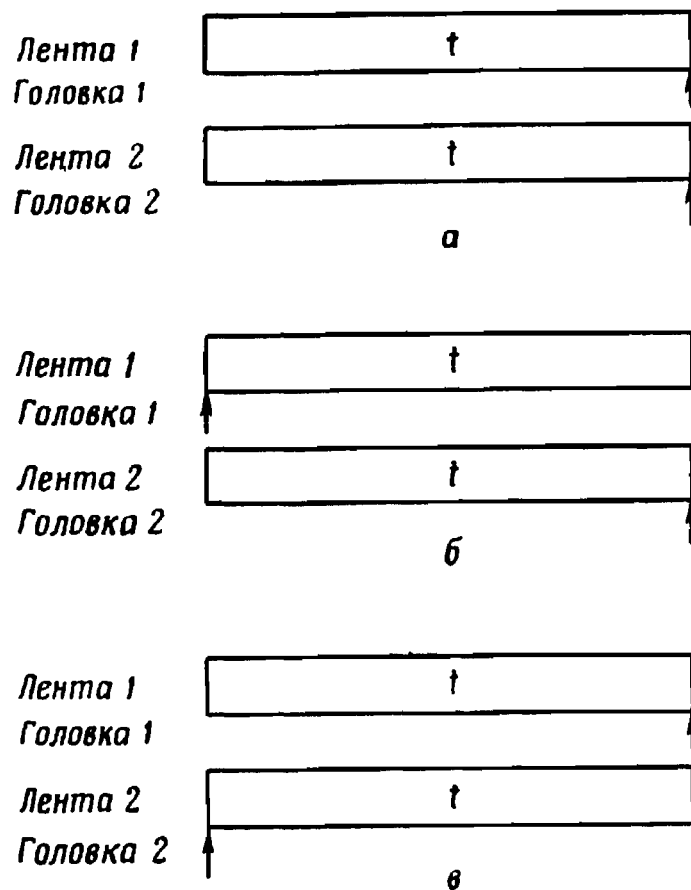
и затем переходит к повторению действий 2-МРВ из леммы 8. Трехленточная машина отвергает всякое слово

$$11t_1110011t_21100 \dots 11t_k1100\beta t,$$

такое, что $L(t) \neq L(t_1)$, — вот для чего она определяет конец входного слова. Детали опять предоставляем читателю.

Мы имеем следующий результат.

Утверждение 2. Существует язык, который определим за время $2L(t)$ и, следовательно, за время $(1 + \epsilon)L(t)$ для любого $\epsilon > 0$, но который не определим в реальное время.



Р и с. 1. а — ленты Т после чтения t_1 ; б — ленты Т после чтения t_{2i-1} ; в — ленты Т после чтения t_{2i} .

4.4. Преобразователи. Результаты разд. 4.1 позволяют нам легко доказать незамкнутость R относительно отображений с помощью различных устройств. Последовательная машина есть 0-МРВ $(Q, \Sigma, \Delta, M, R, q_0)$, такая, что R (выходная функция) отображает $Q \times \Sigma$ в Δ . Вспомним язык P из разд. 4.1. P — язык в алфавите $\Sigma = \{0, 1, \beta\}$. Поэтому, если положить $\Delta = \{a, b\}$, где a и b не принадлежат Σ , очевидно, что Δ^*P есть ЯРВ. Рассмотрим последовательную машину (ПМ) $U = (\{q_0\}, \Sigma \cup \Delta, \Sigma, M, R, q_0)$, где (1) для σ из $\Sigma \cup \Delta$, $M(q_0, \sigma) = q_0$ и где (2) $R(q_0, a) = R(q_0, 0) = 0$; $R(q_0, b) = R(q_0, 1) = 1$, $R(q_0, \beta) = \beta$. Ясно, что $U(\Delta^*P) = \{0, 1\}^*P$, а этот язык, как мы знаем, не принадлежит классу R .

Теорема 10. *Образ языка, определимого в реальное время, при отображении последовательностной машиной может не быть определимым в реальное время.*

Следствие 10.1. *Образ ЯРВ при отображении обобщенной последовательностной машиной не обязательно является ЯРВ.*

Следствие 10.2. *Гомоморфный образ ЯРВ не обязательно является ЯРВ.*

Следует из того, что гомоморфизм — это отображение, определяемое ОПМ с одним состоянием.

Следствие 10.3. *Образ ЯРВ при отображении, осуществляемом преобразователем, работающим в реальном масштабе времени, может не быть ЯРВ.*

Наконец, покажем, что прообраз ЯРВ при отображении недетерминированной последовательностной машиной может не быть ЯРВ. Недетерминированная последовательностная машина есть шестерка $U = (Q, \Sigma, \Delta, M, R, Q_0)$, где Q — конечное множество состояний, Σ — входной, а Δ — выходной алфавиты, M — функция, отображающая $Q \times \Sigma$ в 2^Q , R — отображение $Q \times \Sigma$ в Δ^* и $Q_0 \subseteq Q$ — множество начальных состояний. U действует очевидным способом. Образ $U(x) = \bigcup_{q \in Q_0} R(q, x)$ слова x , если R

распространено на $Q \times \Sigma^*$, есть теперь множество слов, а именно множество, полученное проходом по всем путям, описываемым U (посредством M -отображения) под действием входа x .

Рассмотрим теперь недетерминированную последовательностную машину U , определенную следующим образом: $\Sigma = \{0, 1, \beta\}$,

$$\Delta = \{a, 0, 1, \beta\}, \quad Q_0 = \{q_0, q_1\},$$

M	0	1	β	R	0	1	β
q_0	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_3\}$	q_0	a	a	β
q_1	$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$	q_1	0	1	β
q_2	$\{q_2\}$	$\{q_2\}$	$\{q_3\}$	q_2	a	a	β
q_3	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$	q_3	0	1	β

Язык $L = \{utv\beta t^T : t \in \{0, 1\}^*, u, v \in \{a\}^*; \beta \notin \{0, 1, a\}\}$ есть ЯРВ. Прообраз L при отображении U есть

$$U^{-1}(L) = \{x : U(x) \cap L \neq \emptyset\} = \{xty\beta t^T : x, t, y \in \{0, 1\}^*, \beta \notin \{0, 1, a\}\}.$$

Доказательства теоремы 6 в разд. 4.1 достаточно, чтобы показать, что $U^{-1}(L)$ не содержится в R .

Утверждение 3. *Прообраз ЯРВ при отображении недетерминированной последовательностной машиной может не быть ЯРВ.*

4.5. Производные и частные. В этом разделе мы докажем, что ни производное¹⁾, ни частное ЯРВ по регулярному множеству не определимо в реальное время.

Определение 8. Пусть A и B — языки. **Производное A по B** (обозначается $D_B(A)$) есть множество

$$D_B(A) = \{y: xy \in A \text{ для некоторого } x \in B\}.$$

В терминологии Гинзбурга и Спеньера [3] $D_B(A)$ есть *левое частное A по B* и обозначается там $B \setminus A$.

Вспоминая язык Q из разд. 4.3, мы рассмотрим множество V последовательностей вида

$$t\beta 11t_1110011t_21100 \dots 11t_k1100\beta t,$$

где (1) $t, t_1, \dots, t_k$ принадлежат X_m для некоторого m , а $\beta \notin \{0, 1\}$ и (2) по крайней мере для одного $i \in \{1, \dots, k\}$ $t_i = t^T$.

Лемма 10. *Язык V есть ЯРВ.*

Доказательство. 2-МРВ, которая определяет V , действует почти так же, как 2-МРВ, которая определяет Q (лемма 8), за исключением того, что, встречая второе вхождение β , она сравнивает, совпадает ли конечная цепочка с начальной. Из рис. 1 (б) и (в) видно, что это всегда возможно. Ч. т. д.

Теорема 11. *Производное ЯРВ по регулярному множеству не обязательно определимо в реальное время.*

Доказательство. По лемме 10 V есть ЯРВ, множество $A = \{0, 1\}^*\{\beta\}$ регулярно. Однако $D_A(V) = Q^T$, которое, как мы показали, не определимо в реальное время (теорема 9). Ч. т. д.

Теперь, рассматривая результат разд. 4.1, мы покажем, что частное множества из $\mathbf{R}$ по регулярному множеству не обязательно определимо в реальное время.

Определение 9. Пусть A и M — множества слов. **Частное A по B** (обозначается A/B) есть множество $A/B = \{x: xy \in A \text{ для некоторого } y \in B\}$.

В терминологии Гинзбурга и Спеньера [3] A/B — *правое частное A по B* .

Вспоминая множества P и Σ^*P из разд. 4.1, мы определим R как множество всех лент вида

$$x11t1100y\beta t^T \delta a^L(y)+L(t)+6 \delta a^L(t)+6,$$

¹⁾ Термин «производное» (derivative) был впервые введен Бжозовским (Brzozowski, Derivatives of regular expressions, J. ACM, 11 (1964), 481—494).

где x, y принадлежат Σ^* , $t \in (\Sigma^* \setminus \Sigma^*11\Sigma^*)$, a — элемент Σ , через a^k обозначена цепочка $\underbrace{a \dots a}_{k \text{ раз}}$ и β, δ — символы, не принадлежащие Σ .

Лемма 11. *Язык R определим машиной 2-МРВ.*

Доказательство. Как и в предыдущей лемме, мы неформально опишем работу 2-МРВ T , которая допускает R .

(1) T копирует входные символы на ленте 1, пока она не встретит символ β . Тогда она запоминает следующие входные символы на ленте 2, пока не встретится первое δ . В этот момент головка ленты 1 (соотв. ленты 2) стоит на крайнем правом символе скопированной цепочки $x11t1100y$ (соотв. t^T).

(2) Во время восприятия цепочки $a^{L(y)+L(t)+6}$ T двигает ленты 1 и 2 так, что в момент восприятия второго δ головка ленты 2 находится на самом левом непустом символе ленты 2, а головка ленты 1 находится на некотором символе отрезка $x11t1100y$. Заметим, что если входная лента действительно принадлежит R , то головка ленты 1 должна находиться на самом левом символе отрезка $11t1100y$.

(3) Во время восприятия конечной последовательности $a^{L(t)+6}$ T проверяет, содержит ли лента 2 t^T , где начальный отрезок длины $L(t) + 6$ правее головки ленты 1 есть $11t1100$. T приходит в допускающее состояние, если это последнее условие выполнено.

Ясно, что $A(T) = R$. Ч. т. д.

Рассмотрим регулярное множество $A = \{\delta\}\{a\}^*\{\delta\}\{a\}^*$. Ясно, что $R/A = \Sigma^*P$, которое, как мы показали в теореме 6, не определимо в реальное время. Поскольку R принадлежит R , A — регулярное множество и R/A не принадлежит R , нами доказана следующая теорема.

Теорема 12. *Частное языка, определимого в реальное время, по регулярному множеству может не быть языком, определимым в реальное время.*

4.6. Максимизация. Последней мы рассмотрим операцию максимизации.

Пусть L — язык, тогда $\text{тах } L = \{x: x \in L, \text{ и если } x < y, \text{ то } y \notin L\}$.

Теорема 13. (А. Р. Мейер¹⁾). *Существует ЯРВ L , такой, что $\text{тах } L$ не определим в реальное время.*

¹⁾ Частное сообщение.

Доказательство. Мы предполагаем знакомство с понятиями универсальной машины Тьюринга и арифметизации машин Тьюринга. Рассмотрим арифметизацию машин Тьюринга (МТ), в которой каждое натуральное число x соответствует МТ. Мы предположим, что целые числа закодированы двоичным кодом. Возьмем универсальную машину Тьюринга T и определим язык H . H есть множество всех последовательностей вида $x\beta z$, где

(1) x содержится в $\{1\}\{0, 1\}^*$, β не содержится в $\{0, 1\}$; $z \in \{0\}^*$;

(2) универсальная МТ T обнаруживает за не более чем $L(z) - L(x)$ шагов, что МТ, соответствующая номеру x , закодированному в двоичном коде, останавливается, получая на вход x . Иначе говоря, T при данном x успевает вычислить за не более чем $L(z) - L(x)$ шагов предикат $(\exists n) T(x, x, n)^1$.

Теперь определим язык L :

$$L = \{1\}\{0, 1\}^*\{\beta\} \cup H.$$

H по определению определим машиной 1-МРВ²). 1-МРВ копирует вход на своей рабочей ленте до тех пор, пока не встретит символ β . Затем она двигает обратно свою рабочую ленту по одной клетке на каждый приходящий 0, пока не достигнет левого конца x . С этого момента она имитирует один шаг универсальной МТ T для каждого приходящего на вход 0. 1-МРВ переходит в допускающее состояние, когда она обнаружит, что T остановилась, ответив «ДА» на вход x , и остается в этом состоянии до тех пор, пока не перестает получать символы 0 на вход.

Так как H есть ЯРВ, то по теореме 1 язык L также ЯРВ. Однако легко убедиться в том, что

$$\text{тах } L = \{x\beta: x \in \{1\}\{0, 1\}^*, \beta \notin \{0, 1\} \text{ и } \forall n \neg T(x, x, n)\},$$

что не является даже рекурсивным множеством. Поскольку каждый ЯРВ контекстный и, значит, рекурсивный, мы заключаем, что $\text{тах } L$ не определим в реальное время. Ч. т. д.

4.7. Распространение результатов. В этом разделе мы покажем, что отрицательные результаты, полученные в предыдущих разделах, справедливы и для более широкого класса машин,

¹) Предикат $T(x, y, n)$ означает, что МТ с номером x , примененная к входу y , останавливается за n шагов.

²) Рассматривая язык $H/\{\beta\}\{0\}^*$, можно получить другое доказательство теоремы 12. В самом деле получится более сильный результат, что частное множества из $R(1)$ по регулярному множеству не обязательно содержится в R .

включающего почти все модели, которые могут быть использованы для изучения вычислений в реальное время. Для всякого n класс языков, определяемых n -мерными итеративными сетями конечных автоматов (Коул [1]), не замкнут относительно итерации. Возвратимся снова к отношению $E_k(\text{mod } A)$ (определение 5).

Определение 10. Класс машин $\mathfrak{M}$ есть класс *полиномиально-ограниченных машин*, если для каждой машины M из $\mathfrak{M}$ существуют константы c и n , такие, что для $k \geq 1$ индекс отношения $E_k(\text{mod } A(M))$ не превосходит c^{k^n} .

Мы укажем несколько примеров классов полиномиально-ограниченных машин.

(1) Класс многоленточных машин Тьюринга, работающих в реальном масштабе времени.

(2) Для всякого n класс многоголовочных машин Тьюринга с n -мерными лентами, работающих в реальном масштабе времени.

(3) Для всякого n класс n -мерных итеративных сетей конечных автоматов, работающих в реальном масштабе времени.

Внимательного рассмотрения доказательств разд. 4 достаточно, чтобы доказать теорему 14.

Теорема 14. Пусть $\mathfrak{M}$ — класс полиномиально-ограниченных машин.

(а) Если класс языков $L(\mathfrak{M})$, определяемых $\mathfrak{M}$ -машинами, включает класс языков, определяемых работающими в реальное время PDS-автоматами, то $L(\mathfrak{M})$ не замкнут относительно сцепления даже с регулярным множеством относительно итерации и относительно отображения последовательностной машиной.

(б) Если класс языков, определяемых $\mathfrak{M}$ -машинами, включает класс языков, определяемых 2-MPB-машинами (что необходимо есть частный случай (а)), то $L(\mathfrak{M})$ не замкнут относительно обращения и операций взятия производных и частных.

Можно заметить, что все языки, определяемые с помощью любого класса машин из указанных выше, рекурсивны. Поэтому они не замкнуты относительно максимизации ввиду 4.6.

4.8. Положение R в лингвистической иерархии. В разд. 2 мы отмечали, что R — подкласс класса контекстных языков. Результаты разд. 3 и 4 показывают, что R не сравним с бесконтекстными (КС-) языками; так, существуют КС-языки (например, Σ^*R из разд. 4.1), не являющиеся ЯРВ, и существуют языки, определяемые в реальное время (такие, как $\{a^n b^n c^n: n \geq 1\}$), не являющиеся КС-языками. В этом разделе мы представим пример де-

терминированного КС-языка, не принадлежащего $\mathbf{R}$. Это полностью закончит обоснование рис. 2, на котором показано положение $\mathbf{R}$ в классической лингвистической иерархии.

Пусть L — множество всех последовательностей

$$0^{k_1}10^{k_2}1 \dots 0^{k_{r-1}}10^{k_r}\gamma^s 0^{k_{r-s+1}},$$

где

- (1) $\gamma \notin \{0, 1\}$;
- (2) $r \geq 1, \quad k_i \geq 1 \quad (1 \leq i \leq r)$;
- (3) $1 \leq s \leq r$.

Легко убедиться в том, что L — детерминированный КС-язык. Мы покажем, что L не принадлежит $\mathbf{R}$, рассматривая индекс отношения $E_p(\text{mod } L)$ при росте p .

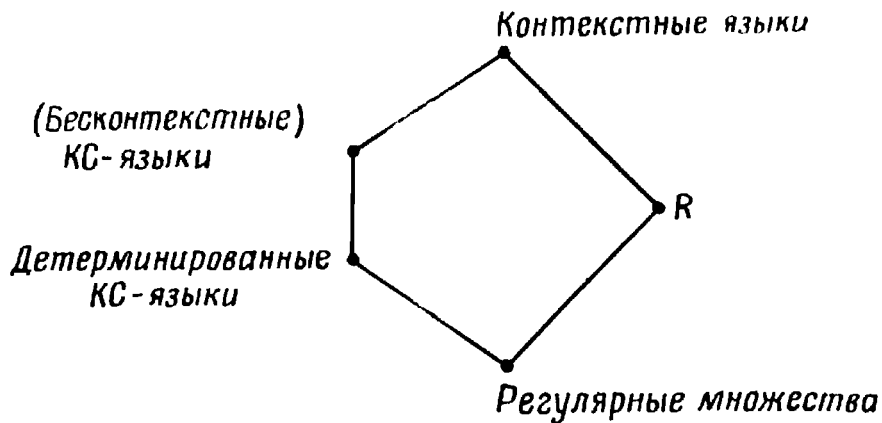


Рис. 2. Положение $\mathbf{R}$ в лингвистической иерархии.

Замечание. Коул [1] и автор [10] независимо нашли другой пример детерминированного КС-языка, не принадлежащего $\mathbf{R}$. Мы рассматриваем L из-за его простоты.

Теорема 15. *Существует детерминированный КС-язык, не определяемый в реальное время.*

Доказательство. Предположим, что язык L определим n -МРВ T с d внутренними состояниями и w рабочими символами.

Рассмотрим подмножества $L_1, L_2, \dots$ языка L , такие, что для всякого целого m L_m есть множество всех последовательностей

$$0^{k_1}10^{k_2}1 \dots 0^{k_{r-1}}10^{k_r}\gamma^s 0^{k_{r-s+1}}$$

из L с $k_i \leq m$ ($1 \leq i \leq r$). Как легко убедиться, число классов эквивалентности отношения $E_{m+r}(\text{mod } L)$ для данного r не меньше чем m^r . Рассмотрим две различные начальные последовательности

$$t_1 = 0^{h_1}10^{h_2}1 \dots 0^{h_r}, \quad 1 \leq h_1, \dots, h_r \leq m,$$

и

$$t_2 = 0^{k_1} 1 0^{k_2} 1 \dots 0^{k_r}, \quad 1 \leq k_1, \dots, k_r \leq m.$$

Тогда ясно, что существует последовательность $u = \gamma^a 0^b$ с $a + b \leq m + r$, такая, что $t_1 u$ принадлежит L_m и, следовательно, L , в то время как $t_2 u$ не принадлежит L_m и, следовательно (по определению L_m), не принадлежит и L . Теперь из леммы 1 мы знаем, что число классов эквивалентности отношения $E_{m+r} \pmod{A(T)}$ не может превзойти $d \cdot \omega^n \cdot \omega^{2n(m+r)}$. Однако легко убедиться, что для достаточно больших r и m $m^r > c^{m+r}$ для любой константы c .

Отсюда следует, что T не может определять L , что противоречит нашему предположению. Ч. т. д.

5. ПРОБЛЕМЫ РАЗРЕШИМОСТИ

Мы вкратце напомним несколько неразрешимых проблем, относящихся к определимости в реальное время. Доказательства легко вытекают из неразрешимости комбинаторной проблемы Поста [7].

Теорема 16. (а) *Проблема: является ли произвольный язык ЯРВ, алгоритмически неразрешима.*

(б) *Следующие проблемы алгоритмически неразрешимы для двух ЯРВ X и Y :*

- (I) *пусто ли X ?*
- (II) *совпадает ли X с Σ^* ?*
- (III) *бесконтекстен ли X ?*
- (IV) *регулярен ли X ?*
- (V) *пересекаются ли X и Y ?*
- (VI) *будет ли $X \subseteq Y$?*
- (VII) *равны ли X и Y ?*

ЛИТЕРАТУРА

1. Cole S. N., Real-time computation by iterative arrays of finite-state machines, Doctoral thesis, Rep. BL-36 by Staff of Harvard Computation Lab. to Bell Telephone Labs., 1964.
2. Ginsburg S., Greibach S. A., Deterministic context free languages, IEEE Conference Record on Switching Circuit Theory and Logical Design, IEEE Pub. 16C13, 1965, 203—220¹⁾.
3. Ginsburg S., Spanier E. H., Quotients of context free languages, J. ACM, 10 (1963), 487—492.

¹⁾ См также статью тех же авторов под тем же названием в Inform. and Contr., 9 (1966), 620—648. — Прим ред.

4. Ginsburg S., Ullian J., Ambiguity in context free languages, *J. ACM*, **13** (1966), 62—89
5. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, **117**, № 5 (1965), 285—306. (Русский перевод: Кибернетический сборник (новая серия), вып. 4, М., 1967.)
6. Kuroda S., Classes of languages and linear-bounded automata, *Inform. and Contr.*, **7** (1964), 207—223.
7. Post E., A variant of a recursively unsolvable problem, *Bull. Amer. Math. Soc.*, **52** (1946), 264—268.
8. Rabin M. O., Real time computation, *Israel J. Math.*, **1** (1963), 203—211. (См. стр. 156—167 настоящего сборника.)
9. Rosenberg A. L., On the nonclosure of the real time definable sets under concatenation, closure, and reversal, IBM Res. Rep. RC-1572, 1966.
10. Rosenberg A. L., A remark on real-time definable languages, IBM Res. Note NC-616, 1966.

МОДЕЛИРОВАНИЕ МНОГОЛЕНТОЧНОЙ МАШИНЫ ТЬЮРИНГА НА ДВУЛЕНТОЧНОЙ ¹⁾

Ф. К. Хенни, Р. Е. Стернс

1. ВВЕДЕНИЕ

Изучение вычислительной способности устройств, подобных вычислительным машинам, было начато Тьюрингом [1], который установил, что механически вычислимые функции являются как раз теми, которые могут быть вычислены устройством с конечным числом состояний и с одной бесконечной лентой, при помощи которой происходит считывание и запись информации. Такое устройство называется машиной Тьюринга. Посредством несущественных изменений способов ввода и вывода информации машина Тьюринга может быть использована для других задач, таких, как генерирование последовательностей и распознавание множеств. Таким образом, функции, последовательности и множества могут быть классифицированы как «вычислимые» или «невычислимые» в зависимости от того, могут ли они быть определены с помощью подходящей машины Тьюринга.

После появления современных быстродействующих вычислительных машин интерес к задачам, связанным с возможностью вычислений, распространился на вопросы, касающиеся «трудности» или «сложности» вычислений. Одним из плодотворных подходов к измерению сложности вычисления является оценка времени, необходимого для этого вычисления. Будем говорить, что, какова бы ни была целочисленная функция $T(n)$, данная функция, множество или последовательность $T(n)$ -вычислимы тогда и только тогда, когда некоторая (соответствующим образом модифицированная) машина Тьюринга за $T(n)$ операций вычисляет n -й член последовательности или (при другой постановке задачи) обрабатывает последовательность длины n . Ниже будут рассмотрены три варианта машины Тьюринга, к которым применяется этот критерий сложности. Все эти модели имеют устройство с конечным числом состояний, управляющее одной или несколькими бесконечными лентами. Заметим, что использование более чем одной ленты не дает машине никаких дополнительных преимуществ, но тем не менее некоторые вычисления могут быть сделаны с меньшим, чем на одноленточной машине,

¹⁾ Hennie F. C., Stearns R. E., Two-tape simulation of multitape Turing machines, *Journal of ACM*, 13, № 4 (1966), 533—546.

числом операций, и, кроме того, применение более чем одной ленты делает машину Тьюринга более похожей на цифровые вычислительные машины.

Основная задача теории вычислительной сложности — это определение того, какие преимущества дает вычисление на k_1 -ленточной машине по сравнению с k_2 -ленточной машиной. Эта задача интересна не только сама по себе, так как от скорости, с которой машина с фиксированным числом лент имитирует последовательность машин с произвольным числом лент, зависит проведение известных доказательств, основанных на канторовском диагональном методе. Такие приложения исследуются нами ниже при рассмотрении конкретных моделей.

Хартманис и Стирнз [2] показали, что одноленточная машина может моделировать работу k -ленточной машины за время, не большее, чем квадрат времени работы k -ленточной машины. Хенни [3] показал, что имеются случаи, в которых доведение числа лент от двух до одной в действительности требует квадрата времени вычисления. Таким образом, моделирование может быть произведено достаточно эффективно только с использованием более чем одной ленты.

Целью данной статьи является описание схемы, при помощи которой двуленточная машина Тьюринга за $n \log n$ операций может имитировать работу многоленточной машины Тьюринга, произведенную за n операций. По сравнению со схемой, использующей одну ленту, эта схема позволяет улучшить соответствующие результаты, получаемые при помощи диагонального метода. Такие приложения будут рассмотрены в последующих разделах.

2. ПРЕДВАРИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

Цель этого раздела — более подробное описание рассматриваемых нами объектов. Однако как определения, так и доказательства по необходимости должны быть в какой-то степени неформальными, так как строгое изложение потребует слишком много места и затруднит понимание простых принципов, на которых основана рассматриваемая конструкция.

Опишем основную модель *многоленточной машины Тьюринга* как вычислительную машину с конечным автоматом в качестве управляющего устройства. Этому управляющему устройству подчинено фиксированное число лент, каждая из которых линейна, не ограничена в обе стороны и разделена на бесконечную последовательность квадратов. Управляющее устройство имеет на каждой ленте одну считывающую головку, которая обзревает один из квадратов данной ленты. Имеется конечное число

различных символов, которые могут быть записаны в квадратах лент. Каждая комбинация считываемых головками символов вместе с состоянием управляющего устройства однозначно определяет машинную операцию. Машинная операция заключается в том, что на каждом квадрате, обозреваемом головкой, печатается символ, ленты независимо друг от друга сдвигаются на один квадрат влево или вправо или же остаются на месте, а также изменяется состояние управляющего устройства. Машина после этого готова выполнить следующую операцию, которая определяется лентами и состоянием управляющего устройства. Если данная операция не требует сдвига лент, изменения символов, записанных на лентах, или изменения состояния, тогда говорят, что машина *остановилась*.

Для того чтобы использовать машину для вычислений, нужно задать машине некоторые значения в качестве входных данных и получить выходные данные. В последующих разделах рассматриваются различные способы реализации входа и выхода, хотя при моделировании они не имеют особого значения, так как для оценок важны лишь основные машинные операции.

Таким образом, задача заключается в моделировании основной модели; ввод входной и вывод выходной информации не имеют значения. Обычно под «моделированием» мы понимаем просто процесс имитации одной за другой операций, осуществляемых данной машиной.

3. МЕТОД МОДЕЛИРОВАНИЯ

Набросок построения. Цель этого раздела — описать метод быстрой имитации поведения данной k -ленточной машины M_k подходящей двуленточной машиной M_2 . Основная единица процесса имитации, называемая шагом, состоит в моделировании действий, которые M_k совершает в течение одного движения. В начале каждого шага имитации M_2 должна иметь на своих лентах запись того, что в настоящий момент имеется на лентах M_k , и помнить положение считывающих головок относительно лент. За время своего шага M_2 должна не только изменить свою запись, отражающую ситуацию на лентах M_k , но и также перегруппировать эту запись таким образом, чтобы мог быть осуществлен удобный доступ к тем символам, которые будут в дальнейшем использоваться M_k .

Одну из лент M_2 будем называть *лентой памяти*. Представим себе, что эта лента разделена на k *горизонтальных дорожек*, каждая из которых соответствует одной ленте M_k . В свою очередь каждая дорожка разделена на верхний и нижний *уровни* так, как показано на рис. 1. На каждом из маленьких

квадратиков может быть записан любой из ленточных символов M_k и некоторый «маркер». Предполагается, что головка, обзоревающая ленту памяти, воспринимает все квадраты одного столбца при выполнении одной операции. Такой специфический способ чтения ленты памяти соответствует традиционной модели машины Тьюринга, так как вектор символов, записанный в данном столбце этой ленты, можно считать отдельным символом расширенного внешнего алфавита.

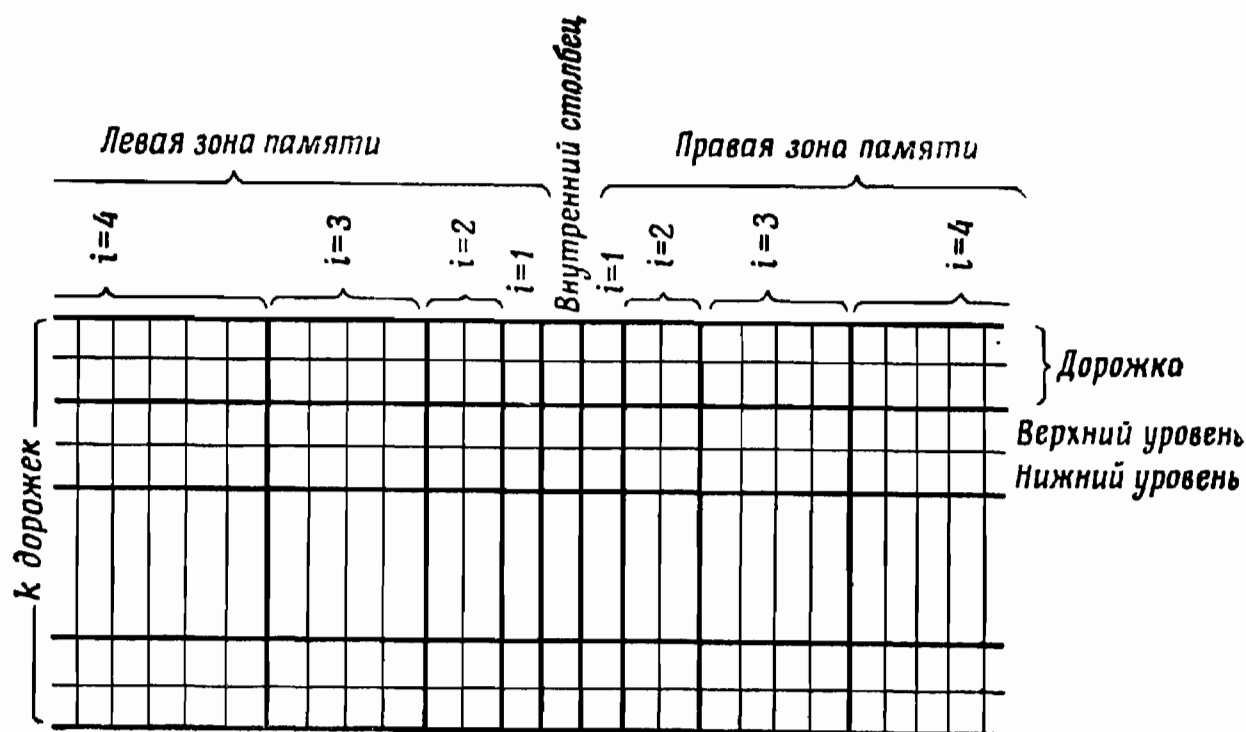


Рис. 1. Лента памяти.

Один из столбцов ленты памяти называется внутренним столбцом. Столбцы слева и справа от внутреннего столбца объединяются в группы, называемые зонами памяти; i -я зона памяти, как показано на рис. 1, содержит ровно 2^{i-1} столбцов. Таким образом, число столбцов в i -й левой (или правой) зоне памяти в точности на 1 больше, чем общее число столбцов в первых $(i-1)$ левых (соответственно правых) зонах памяти. В начале работы границы зон памяти не отмечены; в процессе моделирования M_2 отмечает их специальными символами.

Такая организация ленты памяти может показаться сперва странной, но она дает возможность эффективно провести моделирование M_k . Нижний уровень каждой дорожки используется для запоминания символов, имеющих на лентах M_k . Так, в частности, все символы, считываемые с лент M_k в начале моделирования, размещены во внутреннем столбце. После каждого шага моделирования M_2 перестраивает свои предшествующие записи так, чтобы последующие символы, которые должны быть прочитаны машиной M_k , помещались во внутреннем

столбце. Эта перестройка осуществляется путем воспроизведения символов на второй ленте M_2 и переписывания их затем обратно в нужную зону ленты памяти. Если M_2 сдвигает все символы на каждом шаге, то время, необходимое для имитации n шагов работы M_k , не меньше чем n^2 .

Для того чтобы уменьшить время моделирования, M_2 должна работать таким образом, чтобы в течение шага имитации она сдвигала только небольшой отрезок около внутреннего столбца и оставляла на месте большую часть записанной информации. Такие ограниченные сдвиги повлекут за собой то, что некоторые сдвигаемые отрезки переместятся на место, занятое несдвинутыми символами. В таких случаях, пока не освободится место на нижнем уровне, будет заполняться верхний уровень дорожки. Для каждого такого отрезка избыточных символов на другой стороне внутреннего квадрата должен быть отведен соответствующий промежуток. Эти промежутки остаются свободными, пока не потребуются в дальнейшем.

Количество символов, которые должны быть сдвинуты в течение каждого шага, определяется размерами зон памяти и числом избыточных символов, записанных на верхнем уровне. Размеры зон памяти выбираются таким образом, чтобы в i -й зоне памяти помещался избыток, получающийся в результате перемещения символов на первых $i-1$ зонах памяти.

Число шагов, необходимых для осуществления таких преобразований, близко к минимуму, и имитация может быть проведена достаточно эффективно.

В частности, будет показано, что n операций M_k могут быть промоделированы на M_2 самое большее за $n \log n$ операций.

Описание моделирования. В начале каждого шага моделирования символы, считываемые M_k со своих лент, должны находиться на соответствующих дорожках ленты памяти M_2 во внутреннем столбце. Тогда M_2 для того, чтобы произвести изменения символов, требуемые на этом шаге, и определить направления, в которых должны сдвинуться ленты M_k , необходимо только прочесть внутреннюю колонку. Разумеется, для того чтобы подготовиться к следующему шагу моделирования, M_2 необходимо так перестроить свою запись на ленте памяти, чтобы те символы, которые M_k должна будет считать, попали во внутренний столбец. Это преобразование является основным в моделировании и будет подробно рассмотрено ниже. Так как алгоритм в основном одинаков для каждой из лент M_k , мы ограничимся рассмотрением одной ленты и соответствующей дорожки ленты памяти M_2 .

В каждый момент времени только некоторые квадраты дорожки будут использоваться для хранения символов, записанных на соответствующей ленте M_k .

Будем называть квадраты, в которых хранятся некоторые символы с ленты M_k , полными, в противном случае будем называть квадраты пустыми. В начале каждого шага моделирования ситуация на каждой ленте M_k будет записана на соответствующей дорожке ленты памяти M_2 в соответствии со следующими правилами.

1. На любом уровне каждой зоны памяти квадраты либо все полны, либо все пусты. Заметим, что нужно различать пустые квадраты и квадраты, используемые для хранения пробелов. Если ленты M_k первоначально пусты, то лента памяти M_2 также пуста, но нижние уровни каждой дорожки хранят пробелы, тогда как верхние уровни пусты.

2. На каждой дорожке, если верхний уровень i -й левой (правой) зоны памяти полон, нижний уровень той же зоны должен быть также полон и оба уровня i -й правой (левой) зоны должны быть пустыми. С другой стороны, если нижний уровень i -й левой (правой) зоны пуст, то верхний уровень той же зоны должен быть пуст и оба уровня i -й правой (левой) зоны должны быть полны. Верхний уровень во внутреннем столбце всегда пуст, а нижний уровень, в дальнейшем называемый внутренним квадратом, всегда полон. В начале моделирования верхний уровень каждой зоны будет пуст, а ее нижний уровень — полон.

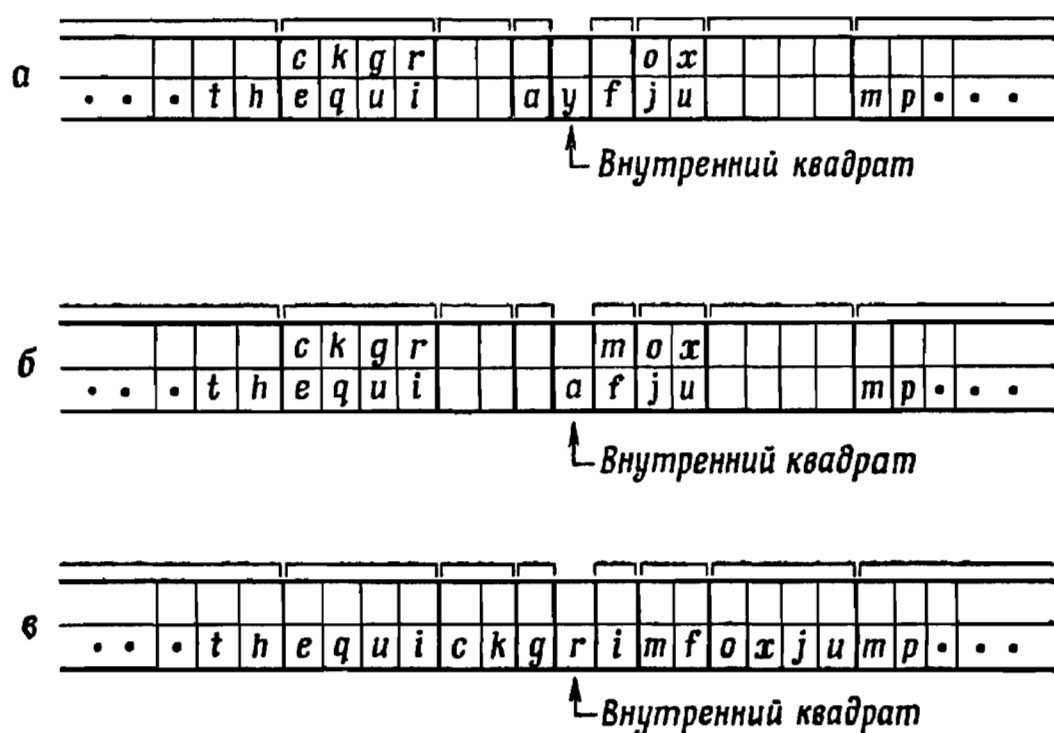
3. Символы на каждой дорожке должны быть записаны так, чтобы соответствующая запись на ленте M_k могла быть получена следующим образом. Начиная с самого левого непустого символа данной дорожки читается слева направо то, что записано в зонах памяти, причем пустые зоны игнорируются. Слева от внутреннего квадрата символы, хранящиеся на нижнем уровне данной зоны, считываются раньше тех, что записаны на верхнем. Справа от внутреннего квадрата этот порядок меняется.

В качестве примера на рис. 2, а изображена типичная конфигурация, которая может быть записана на дорожке ленты памяти. Буквы изображают символы внешнего алфавита M_k , а пустые квадраты изображают пробелы. Читатель может убедиться, что эта конфигурация удовлетворяет правилам 1 и 2. И в соответствии с правилом 3 она изображает запись

... thequickgrayfoxjump ...

на ленте M_k , причем под считывающей головкой M_k находится символ «y».

Предположим, что конфигурация символов, изображенная на рис. 2, а, в начале некоторого шага моделирования записана на одной из дорожек ленты памяти M_2 и что в следующий момент времени M_k печатает вместо символа, который она считывает, символ « t », а также сдвигает ленту вправо. Для моделирования этого действия M_2 должна напечатать во внутреннем квадрате символ « t » и изменить конфигурацию символов на дорожке таким образом, чтобы символ « a » попал во внутренний



Р и с. 2. Два последовательных упорядочивания.

квадрат и удовлетворялись правила 1—3. Это может быть достигнуто, как показано на рис. 2, б, путем сдвига символа « a » во внутренний квадрат и помещения только что напечатанного символа « t » на верхний уровень первой справа зоны памяти.

Предположим теперь, что на следующем шаге вычисления M_k печатает вместо символа, который она считывает, символ « i » и сдвигает ленту еще на один квадрат вправо. В этом случае перестройка конфигурации символов будет осуществляться сложнее. Символ, который должен быть сдвинут во внутренний квадрат, находится на верхнем уровне третьей слева зоны памяти. Просто сдвинуть этот символ во внутренний квадрат, оставив третью зону памяти без изменения, невозможно, так как каждый уровень зоны памяти должен быть или целиком полным, или целиком пустым (правило 1). Наиболее быстрый способ восстановить правильную конфигурацию — это переместить все символы из верхнего уровня третьей зоны памяти в нижние уровни левой первой и второй зон памяти. Размеры зон памяти выбраны таким образом, что три из этих символов

заполняют первую и вторую зоны, а четвертый символ « r » займет внутренний квадрат.

Для того чтобы правило 2 было выполнено, необходимо, чтобы верхние уровни двух первых правых зон памяти были пусты, а нижний уровень третьей правой зоны памяти должен быть заполнен. Однако число символов, записанных на верхних уровнях двух первых правых зон, вместе с символом « i », сдвинутым из внутреннего квадрата, как раз равно числу квадратов в нижнем уровне третьей правой зоны памяти. Таким образом, символы могут быть расположены так, как на рис. 2, в, и такая конфигурация, удовлетворяя правилам 1 и 2, точно отражает ситуацию, которая возникла на рассматриваемой ленте M_k .

Основываясь на примере, изображенном на рис. 2, установим теперь основные правила перегруппировки символов на дорожках ленты памяти M_2 . Допустим, что соответствующая запись уже произведена и что M_2 имитирует сдвиг ленты вправо, тогда правила будут следующими.

4. Найти первый непустой квадрат влево от внутреннего квадрата. Пусть i есть номер левой зоны памяти, содержащей этот квадрат.

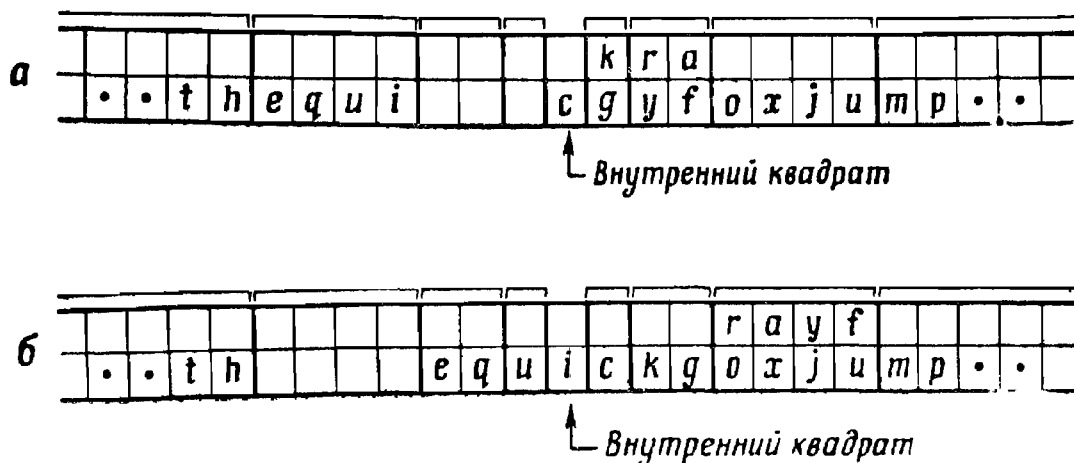
5. Если верхний уровень i -й левой зоны памяти полон, то переписать символы, расположенные на нем (в соответствующем порядке), на нижние уровни первых $(i-1)$ левых зон памяти, а самый правый из них поместить во внутренний квадрат. Если верхний уровень i -й левой зоны памяти пуст, то перегруппировать символы, расположенные на нижнем уровне, тем же способом.

6. Все символы, расположенные на обоих уровнях первых $(i-1)$ правых зонах памяти, вместе с символом, сдвинутым из внутреннего квадрата, переписать (в соответствующем порядке) на нижние уровни первых $(i-1)$ правых зон памяти, причем избыточные символы записываются на i -ю зону. Запись идет на нижний уровень этой зоны, если он свободен; в противном случае используется верхний уровень.

Если M_k сдвигает свою ленту не вправо, а влево, то в этих трех правилах достаточно поменять местами слова «левый» и «правый». В любом случае процесс перестановки символов будем называть упорядочиванием степени i . Переход от конфигурации, изображенной на рис. 2, б, к конфигурации, изображенной на рис. 2, в, является упорядочиванием третьей степени. В данном случае применена первая часть правила 5, так как верхний уровень третьей левой зоны памяти полон. Переход от рис. 2, а к рис. 2, б является упорядочиванием первой степени, в котором применена вторая часть правила 5. Рис. 3 иллюстрирует менее тривиальное приложение правила 5 для того

случая, когда верхний уровень пуст. На этом рисунке изображено упорядочивание третьей степени. Мы считаем очевидным, что данная процедура работает, т. е. происходит упорядочивание конфигурации, удовлетворяющей правилам 1—3, и, таким образом, применимы правила 4—6. Читатель может легко проверить это сам на примерах.

Напомним, что мы рассматривали только одну из лент M_k и соответствующую ей дорожку ленты памяти M_2 . На каждом шаге имитации M_2 должна упорядочить каждую дорожку своей



Р и с. 3. Упорядочивание третьей степени.

ленты памяти и такое упорядочивание определяется правилами, данными выше. Конечно, упорядочивания, выполняемые в данный момент на различных дорожках, не обязательно будут иметь одну и ту же степень. Более того, обычно они имеют разную степень, так как ленты M_k могут последовательно сдвигаться в разные стороны.

Реализация процедуры упорядочивания. Мы рассмотрели процедуру упорядочивания и сейчас покажем, как эта процедура реализуется подходящей двуленточной машиной Тьюринга M_2 . Предположим, что во время выполнения данной операции M_k происходит сдвиг ленты вправо. Машина M_2 печатает символ во внутреннем квадрате и сдвигает головку своей ленты памяти в левую сторону до тех пор, пока под головкой не будет находиться квадрат, в котором записан символ из внешнего алфавита M_k . Когда головка ленты памяти проходит некоторый квадрат во время своего движения налево, на второй, или копирующей, ленте M_2 отмечается один квадрат. Таким образом, когда головка ленты памяти достигает требуемого символа, число квадратов, отмеченных на копирующей ленте, будет как раз равно 2^{i-1} , где i есть номер зоны памяти, содержащей этот символ.

Далее M_2 начинает переписывать символы с соответствующего уровня i -й зоны памяти на отмеченные квадраты копирующей ленты. Когда эти отмеченные квадраты будут заполнены, головка ленты памяти дойдет до левого края i -й зоны. Если этот край зоны памяти предварительно не отмечен, то ставится специальный маркер. Далее головка двигается в обратную сторону, пока она не вернется к левому краю $(i-1)$ -й зоны памяти, и начинается перепись скопированных символов на нижние уровни первых $i-1$ зон. Во внутренний квадрат будет занесен последний оставшийся символ.

Затем M_2 переписывает содержимое первых $i-1$ правых зон памяти на копирующую ленту. Разумеется, для того чтобы M_2 переписала эти символы в правильном порядке, она должна распознавать границы зон памяти. Заметим, что границы первых $i-1$ зон памяти с обеих сторон внутреннего квадрата должны быть отмечены в результате предыдущих упорядочиваний более низких степеней. После того как содержимое первых $i-1$ правых зон памяти размещено на копирующей ленте, M_2 начинает переписывать эту информацию на нижние уровни первых $i-1$ правых зон памяти и на соответствующий уровень i -й зоны. Машина распознает i -ю зону как первую зону с пустым верхним уровнем. Когда все символы переписаны, достигается правая граница i -й зоны памяти, и M_2 отмечает эту границу, если она уже не была отмечена. Таким образом, выполнение упорядочивания i -й степени обеспечивает разметку границ i -й зоны памяти. Так как упорядочиванию i -й степени должны предшествовать упорядочивания всех меньших степеней (см. лемму 2), на ленте уже будут стоять маркеры, отмечающие границы зон памяти с меньшими номерами, и, таким образом, перепись символов может быть произведена, как показано выше.

Далее головка M_2 возвращается к внутреннему квадрату и очищает отмеченные квадраты на копирующей ленте. Теперь упорядочивание закончено, и машина начинает упорядочивать следующую дорожку ленты или переходит к следующему шагу имитации. Заметим, что в процессе выполнения упорядочивания i -й степени M_2 не нужно обрабатывать квадраты ленты, расположенные за i -ми зонами памяти. Кроме того, весь процесс состоит из некоторого числа движений головок внутри зон памяти и соответствующего отрезка копирующей ленты. Таким образом, число движений головок, необходимое для выполнения упорядочивания i -й степени, пропорционально числу квадратов в первых i зонах памяти с обеих сторон внутреннего квадрата.

Время имитации. Вернемся теперь к задаче оценки времени, необходимого M_2 для имитации n операций вычисления,

производимого M_k . Будет показано, в частности, что это время пропорционально $n \log n$. Это будет сделано путем доказательства нескольких простых лемм, в каждой из которых рассматриваются упорядочивания, производимые на одной дорожке.

Лемма 1. *Число операций M_2 , необходимое для выполнения упорядочивания i -й степени, меньше чем $\beta 2^i$, где β есть константа, не зависящая от i .*

Доказательство. Как уже было отмечено, число операций, необходимое для выполнения упорядочивания i -й степени, самое большее пропорционально числу квадратов, содержащихся в первых i зонах памяти с обеих сторон внутреннего квадрата. Но это число квадратов как раз равно

$$1 + 2 \sum_{j=1}^i 2^j = 2^{i+2} - 1 < 4 \cdot 2^i,$$

и, таким образом, общее время пропорционально 2^i .

Лемма 2. *Перед первым упорядочиванием степени, большей чем i , и между двумя любыми упорядочиваниями степеней, больших чем i , должно быть произведено по крайней мере одно упорядочивание степени i .*

Доказательство. В результате каждого упорядочивания степени, большей чем i , верхние уровни каждой из двух i -х зон памяти становятся пустыми. Эти зоны в начале имитации также являются пустыми. Однако перед упорядочиванием степени, большей чем i , верхний уровень одной из двух i -х зон памяти должен быть полон, так как иначе должно выполняться упорядочивание меньшей или равной i степени в соответствии с правилами 4 и 2. Но верхний уровень i -й зоны памяти может быть заполнен только при выполнении упорядочивания i -й степени.

Лемма 3. *Перед первым упорядочиванием степени, большей чем i , и между двумя любыми упорядочиваниями степеней, больших чем i , должны быть произведены по меньшей мере $2^{i-1} - 1$ упорядочиваний степеней, меньших или равных i .*

Доказательство. Проведем доказательство индукцией по i . Для $i = 1$ лемма очевидна. Допустим, что утверждение справедливо для всех i , меньших чем некоторое целое m . Рассмотрим два упорядочивания c_1 и c_2 степеней, больших чем m . По лемме 2 между c_1 и c_2 должно выполняться хотя бы одно упорядочивание c_3 степени m . Но по предположению между c_1 и c_3 должно быть произведено по меньшей мере $2^{m-2} - 1$ упорядочивания и между c_3 и c_2 найдется по меньшей мере $2^{m-2} - 1$

упорядочиваний. Таким образом, общее число упорядочиваний между c_1 и c_2 по меньшей мере равно $2^{m-1} - 1$. Так как подобное рассуждение проходит, если c_2 есть первое упорядочивание степени, большей чем i , лемма доказана.

Лемма 4. Пусть n_i есть число упорядочиваний степени i , выполняемых в процессе имитации первых n шагов работы M_h . Тогда $n_i \leq n/2^{i-2}$.

Доказательство. Перед первым упорядочиванием степени i и между любыми двумя последовательными упорядочиваниями степени i должны выполняться по лемме 3 по меньшей мере $2^{i-2} - 1$ упорядочиваний меньшей степени. Следовательно, самое большое одно упорядочивание из 2^{i-2} последовательных упорядочиваний может быть степени i , и, таким образом, из n последовательных упорядочиваний самое большое $n/2^{i-2}$ могут быть степени i .

Лемма 5. $n_i = 0$ для $i > 2 + \log_2 n$.

Доказательство. Если $i > 2 + \log_2 n$, то по лемме 4

$$n_i \leq \frac{n}{2^{i-2}} < \frac{n}{2^{\log_2 n}} = 1.$$

Так как n_i должно быть целым, то $n_i = 0$.

Лемма 6. Общее число операций, необходимых M_2 для выполнения всех упорядочиваний на одной дорожке в процессе моделирования первых n операций M_h , не превосходит $4\beta n(2 + \log_2 n)$.

Доказательство. Время, необходимое M_2 , есть $T = \sum_{i=1}^{\infty} t_i n_i$, где t_i — время, необходимое для упорядочивания i -й степени. Но в соответствии с леммой 5 моделирование проводится только до $i = \log_2 n + 2$. Далее, применяя леммы 1 и 4, имеем

$$T = \sum_{i=1}^{\log n + 2} t_i n_i \leq \sum_{i=1}^{\log n + 2} \beta 2^i n_i \leq \sum_{i=1}^{\log n + 2} \beta \frac{2^i n}{2^{i-2}} = 4\beta n (\log n + 2),$$

что завершает доказательство.

Наконец, рассмотрев упорядочивания, выполняемые на всех дорожках, докажем следующую теорему:

Теорема 1. Число операций M_2 , необходимое для имитации n операций M_h , не превосходит $\alpha n \log_2 n$ для $n > 1$, где α есть константа, не зависящая от n .

Доказательство. Произведение максимального числа операций, выполняемых на каждой дорожке, на общее число дорожек есть

$$4\beta kn (\log_2 n + 2) \leq 12\beta kn \log_2 n = \alpha n \log_2 n.$$

Можно показать, что существует константа α' , такая, что в худшем случае число операций M_2 , необходимых для имитации n операций M_k , равно самое меньшее $\alpha' n \log_2 n$. Таким образом, результаты теоремы 1 являются лучшими из тех, что можно получить изложенным методом. Читатель может проверить, что наихудшая ситуация возникает в том случае, когда некоторая лента M_k сдвигается в одном и том же направлении на каждом шаге вычисления.

4. ГЕНЕРАТОР ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Генератор последовательностей — это многоленточная машина Тьюринга, у которой с некоторыми состояниями управляющего устройства связаны выходные символы. Он начинает свою работу в некотором выделенном начальном состоянии и с пустыми лентами. В процессе своей работы он иногда попадает в состояния, с которыми связаны выходные символы. Последовательность (бесконечная) этих выходных символов называется **порождаемой** этой машиной последовательностью. Бесконечная последовательность w называется **$T(n)$ -вычислимой** тогда и только тогда, когда существует некоторый генератор последовательностей M , который выдает n -й член w за $T(n)$ или меньше операций. В таком случае говорят, что машина **работает за время $T(n)$** .

Теорема 2. Если последовательность w порождается k -ленточной машиной за время $T(n)$ и $T(n) \geq n$, то эта последовательность может быть порождена двуленточной машиной за время $T(n) \log_2 T(n)$.

Доказательство. Было показано (теорема 1), что двуленточная машина имитирует данную k -ленточную машину за время $\alpha T(n) \log_2 T(n)$, где α есть некоторая константа. Машина M_2 затем может быть преобразована в машину M'_2 , которая за каждую операцию выполняет α или больше операций машины M_2 . Такое ускорение вычисления подробно рассмотрено Хартманисом и Стирнзом [2], и в результате время вычисления оценивается следующим образом:

$$T'(n) \leq \max [n, T(n) \log T(n)].$$

Но если $T(n) \geq n$ и $n > 1$, то имеем

$$T'(n) \leq T(n) \log T(n),$$

что завершает доказательство.

Монотонно возрастающая функция $U(n)$ называется **вычислимой в реальное время** [4] тогда и только тогда, когда существует генератор последовательностей, который на j -й операции печатает единицу, если $j = U(i)$ для некоторого i , и печатает нуль в противном случае. Другими словами, $U(n)$ вычислима в реальное время, если ее характеристическая последовательность является n -вычислимой. Теперь мы получим первый результат, не связанный с числом лент.

Теорема 3. Если $U(n)$ — функция, вычисляемая в реальное время, а $T(n)$ — вычисляемая функция и

$$\inf_{n \rightarrow \infty} \frac{T(n) \log T(n)}{U(n)} = 0,$$

то существует бинарная последовательность ω , которая $U(n)$ -вычислима, но не $T(n)$ -вычислима.

Доказательство. В основном доказательство такое же, как доказательство теоремы 9 в работе Хартманиса и Стирнза [2], но «закон квадрата», используемый там, заменяется нашей теоремой 2. Основная идея доказательства — это порождение последовательности ω генератором, который работает за время $U(n)$ и вычисляет достаточное количество членов каждой $T(n)$ -вычислимой бинарной последовательности ω' для того, чтобы напечатать по крайней мере один член последовательности ω , отличный от соответствующего члена последовательности ω' . Рассмотрим сейчас только основные этапы доказательства.

Пусть M_i есть двуленточный генератор последовательностей с бинарным выходом, имеющий номер i при некотором перечислении всех таких машин Тьюринга. Далее построим «диагональный» генератор последовательностей D , который должен решить три задачи. Одна из этих задач заключается в имитации одной за другой начальных частей вычислений, выполняемых $M_1, M_2, \dots$, и т. д. Второй задачей является порождение в реальное время характеристической последовательности функции $U(n)$. Третьей является порождение (не обязательно в реальное время) характеристической последовательности функции $T(n) \log T(n)$.

Так как D имитирует каждую машину M_i , то она следит за числом операций, необходимых M_i для порождения выходных символов. Число этих операций сравнивается с вычисленным значением $T(n) \log T(n)$ для того, чтобы проверить, работает

ли M_i за время, ограниченное $T(n) \log T(n)$. Тот факт, что эта проверка не может не отставать от имитации M_i , не имеет значения. Важно то, что каждая M_i , которая не укладывается за время $T(n) \log T(n)$, может быть в конечном счете обнаружена и исключена из дальнейшей работы.

Так как машина D должна работать за время $U(n)$, то она строится таким образом, чтобы печатать выходные символы в соответствии с единицами характеристической последовательности функции $U(n)$. В момент порождения j -й единицы возможны два случая. Во-первых, D может нехватить времени для вычисления j -го выходного символа. В этом случае D печатает некоторый произвольный символ, скажем 0. Вторая возможная ситуация заключается в том, что D уже определила j -й выходной символ моделируемой машины. В этом случае D печатает символ, дополнительный к определенному ею выходному символу моделируемой машины.

Имитация каждой машины M_i продолжается до тех пор, пока не возникнет одна из следующих ситуаций:

(а) D определяет, что M_i не укладывается за время $T(n) \log T(n)$;

(б) D вычисляет j -й выходной символ машины M_i за время $U(j)$, причем $U(j)$ — время, прошедшее с начала работы D .

Заметим, что если M_i укладывается за время $T(n) \log T(n)$, то в конечном счете должна возникнуть ситуация (б). Действительно, предположим, что имитация машины M_i начинается после того, как уже произведено B_i операций, и что самое большее C_i операций необходимо для имитации одной операции M_i . Тогда, так как

$$\inf_{n \rightarrow \infty} \frac{T(n) \log T(n)}{U(n)} = 0,$$

должно существовать некоторое j , такое, что

$$B_i + C_i T(j) \log T(j) < U(j).$$

При возникновении ситуации (а) или (б) D прекращает имитацию M_i и начинается имитация M_{i+1} .

Пусть теперь w' — некоторая бинарная последовательность, порождаемая за время $T(n)$. В силу теоремы 2 эта последовательность w' может быть порождена двуленточной машиной M за время $T(n) \log T(n)$. В процессе своей работы D в конце концов имитирует машину M и печатает в качестве своего j -го выходного символа символ, дополнительный к j -му выходному символу машины M . Таким образом, последовательность w , порождаемая D , отличается по крайней мере одним символом от последовательности w' , что завершает доказательство.

5. МАШИНА ТЬЮРИНГА СО ВХОДОМ

Так же как и генератор последовательностей, *машина Тьюринга со входом* является многоленточной машиной Тьюринга, выходы которой связаны с состояниями управляющего устройства. Кроме того, машина со входом имеет специфический ввод входной информации. Машина начинает работать в фиксированном начальном состоянии и с пустыми лентами. После восприятия входного символа она производит некоторое число операций, пока не попадает в состояние, с которым связан выход. После этого машина готова воспринять следующий входной символ, и процесс повторяется. Говорят, что конечная последовательность входных символов *допускается* машиной, если после ее переработки машина попадает в состояние с выходом 1. Множество R конечных последовательностей $T(n)$ -распознаваемо (машиной со входом) тогда и только тогда, когда существует машина Тьюринга со входом M , которая перерабатывает любую входную последовательность длины n за $T(n)$ операций, и R является множеством конечных последовательностей, допускаемых M .

Теорема 4. Если множество R распознается k -ленточной машиной Тьюринга со входом за время $T(n)$, то оно распознается двуленточной машиной со входом за время $T(n) \log T(n)$.

Доказательство. Доказательство по существу не отличается от доказательства теоремы 2.

Теорема 5. Если $U(n)$ — вычислимая в реальное время функция, а $T(n)$ — вычислимая функция и

$$\inf_{n \rightarrow \infty} \frac{T(n) \log T(n)}{U(n)} = 0,$$

то существует множество R , которое $U(n)$ -распознаваемо, но не $T(n)$ -распознаваемо.

Доказательство. Пусть w — бесконечная последовательность из теоремы 3. Определим R как множество входных последовательностей длины n для всех n , таких, что n -й член w есть единица. Очевидно, что R является $U(n)$ -распознаваемым. С другой стороны, легко видеть, что « $T(n)$ -распознаватель» может быть преобразован в генератор последовательностей w , что противоречит теореме 3.

6. МАШИНА ТЬЮРИНГА С ЗАПИСЬЮ НА ЛЕНТЕ

Машина Тьюринга с записью на ленте — это многоленточная машина Тьюринга с двоичным выходом, связанным с некоторыми состояниями, кроме того, одна из лент машины рассма-

тривается как входная лента. Машина начинает работать в фиксированном начальном состоянии, а на входной ленте записано некоторое слово. Это слово ограничено с обеих сторон пустыми квадратами, и его самый левый символ находится под считывающей головкой. При обработке каждого входного слова машина в конце концов попадает в состояние (заключительное), с которым связан выход. Говорят, что входное слово допускается в том и только в том случае, когда этот выход есть 1. Множество R конечных слов $T(n)$ -распознаваемо (машиной с записью на ленте) тогда и только тогда, когда существует машина Тьюринга с записью на ленте, которая обрабатывает любое входное слово длины n за $T(n)$ операций и допускает в точности слова из R .

Теорема 6. *Если множество R распознается k -ленточной машиной Тьюринга с записью на ленте за время $T(n)$, то оно может распознаваться двуленточной машиной с записью на ленте за время $T(n) \log T(n)$.*

Доказательство. Опять доказательство по существу не отличается от доказательства теоремы 2.

Теорема 7. *Если $U(n)$ — вычислимая в реальное время функция, то существует множество конечных последовательностей R , которое $U(n)$ -распознаваемо машиной с записью на ленте, но не $T(n)$ -распознаваемо, где функция $T(n)$ такова, что*

$$\inf_{n \rightarrow \infty} \frac{T(n) \log T(n)}{U(n)} = 0.$$

Доказательство. Построим машину D с записью на ленте, которая имеет входной алфавит $\{0, 1\}$ и распознает множество R за время $U(n)$. Эта машина интерпретирует каждое входное слово как двоичное представление целого числа. Основной частью работы D является моделирование вычисления, производимого i -й двуленточной машиной Тьюринга с записью на ленте при обработке данного входного слова. Две ленты D выделяются для того, чтобы использовать их для записи ситуаций на лентах моделируемых машин M_i .

Перед началом работы D измеряет длину входного слова и отмечает ее на вспомогательной ленте. Затем это слово, подходящим образом закодированное, копируется на соответствующую ленту. Далее происходит подготовка к имитации машины M_i . И наконец начинается имитация вычисления, выполняемого M_i при обработке данного входного слова.

Машина D порождает характеристическую последовательность функции $U(n)$ и сравнивает число единиц в этой после-

довательности с длиной входного слова, отмеченной на вспомогательной ленте. Как только число единиц в порождаемой последовательности станет равным длине данного входного слова, D останавливается, и, таким образом, ее работа укладывается за время $U(n)$. Если в этот момент времени D уже промоделировала работу M_i и определила, допустимо ли данное входное слово, то она печатает символ, дополнительный к выходному символу M_i . Если же D не закончила имитацию работы M_i (т. е. имитация M_i не укладывается за время $U(n)$), то печатается некоторый произвольный символ, например 0.

Так как D обязательно останавливается независимо от данного входного слова, то она распознает некоторое множество входных слов, которое мы обозначим R . По построению R является $U(n)$ -распознаваемым. Покажем теперь, что R не $T(n)$ -распознаваемо. Действительно, предположим, что R является $T(n)$ -распознаваемым некоторой многоленточной машиной Тьюринга. Тогда в соответствии с теоремой 6 оно должно быть $T(n) \log T(n)$ -распознаваемо некоторой двуленточной машиной Тьюринга M_k . Если целое число k , представленное в двоичной форме, подается машине M_k в качестве входного слова длины n , то вычисление потребует самое большее $T(n) \log T(n)$ операций. Оценим теперь число шагов, необходимое для имитации того же самого вычисления на машине D . Определение длины входного слова потребует самое большее n операций, а переписывание входного слова требует не больше чем αn операций, где α есть некоторая константа. Обозначим через B_k число операций, необходимых для того, чтобы подготовиться к имитации, и через C_k максимальное число операций, которое необходимо D для имитации одной операции M_k . Тогда общее число операций, которое необходимо D для имитации работы, совершаемой M_k , равно

$$n + \alpha n + B_k + C_k T(n) \log T(n).$$

Эта величина должна быть $\leq U(n)$ для того, чтобы D успела завершить имитацию. Но, так как между $U(n)$ и $T(n)$ имеется определенная в условии связь, всегда можно выбрать n достаточно большим, чтобы

$$n + \alpha n + B_k + C_k T(n) \log T(n) < U(n).$$

Входное слово с длиной, равной или превосходящей эту критическую величину n , может быть получено просто добавлением достаточного количества нулей к левому краю двоичного разложения k . Результаты обработки этого слова машинами D и M_k различны, и, следовательно, они не могут распознавать одно и то же множество. Таким образом, установлено, что R нераспознаваемо за время $T(n)$, и доказательство завершено.

Основные теоремы 3, 5 и 7 могут быть резюмированы следующим образом. Если функция $T(n)$ вычислима и $U(n)$ вычислима в реальное время, причем $U(n)$ растет быстрее, чем произведение некоторой константы на $T(n)\log T(n)$, то существует вычислительная процедура, которая может быть произведена за время $U(n)$, но не за время $T(n)$.

ЛИТЕРАТУРА

- 1 Turing A. M., On computable numbers, with an application to the Entscheidungsproblem, *Proc. London Math. Soc.*, 2, 42 (1936—1937), 230—265; Correction, *ibid.*, 43 (1937), 544—546.
- 2 Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, 117 (May 1965), 285—306. (Русский перевод: Хартманис Дж., Стирнз Р. Е., О вычислительной сложности алгоритмов, Кибернетический сборник (новая серия), вып. 4, «Мир», 1967, стр. 57—85.)
- 3 Hennie F. C., One-tape, off-line Turing machine computations, *Inform. and Contr.*, 8, № 6 (Dec. 1965), 553—578. (См. стр. 223—248 настоящего сборника.)
- 4 Yamada H., Real-time computation and recursive functions not real-time computable, *IRE Trans. EC-11* (1962), 753—760. (См. стр. 139—155 настоящего сборника.)

МЕТОДЫ ПЕРЕВОДА И СЛОЖНОСТЬ ВЫЧИСЛЕНИЙ ¹⁾

С. С. Раби, П. К. Фишер

ПРЕДВАРИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

Эта статья продолжает исследование (начало которому положили Хартманис и Стирнз [1]) вычислительной сложности двоичных последовательностей, порождаемых многоленточными машинами Тьюринга. Многоленточная машина Тьюринга состоит из управляющего устройства (процессора), некоторого числа линейных лент, каждая из которых не ограничена в обе стороны и разбита на бесконечное множество ячеек (квадратов — squares), и головок на каждой из лент (головки могут читать и писать в обозреваемой ячейке соответствующей ленты; эта ячейка находится «под» головкой). На ленте могут быть символы лишь из конечного множества, а управляющее устройство может находиться только в конечном числе внутренних состояний. *Машинная операция* полностью определяется состоянием процессора и символами, находящимися под головками на лентах, и состоит в печатании символов в каждой из ячеек под головками, сдвигании каждой из лент самое большее на одну ячейку в любом направлении и изменении состояния процессора. Исходя из некоторого начального состояния и набора пустых лент, машина порождает двоичную последовательность на специальной односторонней выходной ленте. Пусть $T(n)$ является строго возрастающей монотонной функцией неотрицательных целых чисел с такими же значениями; мы будем называть двоичную последовательность

$$\alpha = (\alpha_i) = \alpha_0 \alpha_1 \alpha_2 \dots$$

T-вычислимой ($\alpha \in S_T$) тогда и только тогда, когда существует многоленточная машина Тьюринга, порождающая α так, что α_n печатается не более чем за $T(n)$ операций. (В обозначениях мы следуем Хартманису и Стирнзу [1].) Темой этой статьи является сложность классов S_T , где T — вычисляемая монотонная функция.

Назовем *характеристической последовательностью* множества A неотрицательных целых чисел двоичную последователь-

¹⁾ Ruby S. S., Fischer P. C., Translational methods and computational complexity, IEEE Conf. Rec. VI Ann. Symp. on «Circuit theory and logic design», 1965, pp. 173—178.

ность $\alpha = \alpha_0\alpha_1\alpha_2 \dots$, такую, что для всех n $\alpha_n = 1$ тогда и только тогда, когда $n \in A$. Можно заметить, что существует взаимно однозначное отображение φ множества вычислимых монотонных функций в множество вычислимых двоичных последовательностей, задаваемое посредством $\varphi(T) = \alpha$ тогда и только тогда, когда α является характеристической последовательностью области значений T . (Это отображение не будет «на», поскольку каждая последовательность, связанная с монотонной функцией, содержит бесконечное число единиц.)

Имея в виду это соответствие, мы назовем монотонную функцию $T(n)$ ***U(n)-вычислимой функцией*** или $T \in F_U$ тогда и только тогда, когда связанная с ней последовательность U -вычислима, т. е. $\varphi(T) \in S_U$.

Каждой многоленточной машине Тьюринга можно сопоставить $T \in F_n$ (т. е. F_U , где $U(n) = n$), где $T(n)$ выражает число операций машины до того, как будет напечатана n -я выходная цифра. По этой и другой причинам элементы F_n называются ***функциями, вычислимыми в реальное время*** [3]. Аналогично элементы S_n называются ***последовательностями, вычислимыми в реальное время***.

Далее будет использован следующий результат Ямада [3]:

- 1) для целых $k > 0$ функции n^k , $(k+1)^n$, $n! \in F_n$;
- 2) если $T, U \in F_n$, то функции $T(n) + U(n)$, $T(n) \cdot U(n)$, $k \cdot T(n)$, $T(U(n))$, $T(n)^{U(n)} \in F_n$.

S_T -ИЕРАРХИЯ

Хартманис и Стирнз [1] показали, что если $c > 0$ — постоянная, то для всех T

$$S_T = S_{c \cdot T}.$$

Другой их основной результат состоял в том, что для любой вычислимой монотонной функции T множество S_T является рекурсивно перечислимым. Поскольку множество всех бесконечных вычислимых двоичных последовательностей не является рекурсивно перечислимым, то отсюда следует, что существует бесконечно много различных классов сложности. Хартманис и Стирнз показали далее, что бесконечное частично упорядоченное множество классов сложности имеет бесконечное линейное упорядоченное подмножество. Этот результат был получен с использованием диагонального метода и того факта, что вычисление, производимое за $T(n)$ операций на некоторой многоленточной машине, можно осуществить за $(T(n))^2$ операций на одноленточной машине Тьюринга. Точная формулировка этого результата такова.

Теорема (Хартманис и Стирнз). Если T и U суть функции, вычислимые в реальное время, и

$$\inf_{n \rightarrow \infty} \frac{(T(n))^2}{U(n)} = 0,$$

то

$$S_U - S_T \neq \emptyset$$

(т. е. класс $S_U - S_T$ непуст).

Следовательно, $S_n, S_{n^3}, S_{n^7}, S_{n^{15}}, \dots, S_{n^{2^k-1}}, \dots$ образуют (относительно включения) бесконечное линейное упорядоченное множество классов сложности.

Вопрос о том, верно ли, например, что S_{n^2} отличается от S_n , был решен Хенни и Стирнзом [2]. Они показали, что вычисление, требующее $T(n)$ операций на многоленточной машине Тьюринга, можно осуществить на двуленточной машине Тьюринга за $T(n) \log_2 T(n)$ операций. Тогда, применяя диагональный метод, можно доказать теорему.

Теорема (Хенни и Стирнз). Если T и U — функции, вычислимые в реальное время, и

$$\inf_{n \rightarrow \infty} \frac{T(n) \log_2 T(n)}{U(n)} = 0, \quad \text{то} \quad S_U - S_T \neq \emptyset.$$

Таким образом, для $T \in F_n$ $S_{T^{1+\varepsilon}}$ строго содержит S_T и частично упорядоченное множество классов сложности содержит плотное подмножество.

К сожалению, предположение Хартманиса и Стирнза о том, что $\inf_{n \rightarrow \infty} \frac{T(n)}{U(n)} = 0$ влечет $S_U - S_T \neq \emptyset$, остается недоказанным.

Очень сильный критерий $T \log_2 T$ Хенни и Стирнза не позволяет решить вопрос: верно ли, что $S_{n^2 n} \neq S_{2^n}$? Цель настоящей заметки — предложить другой подход к изучению структуры классов сложности S_U и F_U , который сводит вопросы о некоторых классах сложности к вопросам о других, быть может, более «доступных» классах.

ПОДХОД, ОСНОВАННЫЙ НА ПЕРЕВОДЕ

Следующая лемма интересна сама по себе и необходима для некоторых теорем, приводимых ниже.

Лемма. Если $T \in F_n$ и $\alpha \in S_T$, то существует многоленточная машина Тьюринга, вычисляющая $\alpha = \alpha_0 \alpha_1 \alpha_2 \dots$ в точности за время $T(n)$.

Доказательство. По теореме «умножения на постоянную» Хартманиса и Стирнза существует машина Тьюринга, вычисляющая α так, что α_n печатается за время, меньшее чем $T(n)/2$. Эта машина, счетчик реального времени для T , счетчик реального времени для $T/2$ и три дополнительные ленты A_1 , A_2 и A_3 комбинируются для следующей процедуры. Начинает работу $T/2$ -машина, вычисляющая α . Все элементы α , вычисленные ко времени $T(1)/2$, записываются по порядку на ленте A_1 . В момент $T(1)/2$ (или $(T(1) + 1)/2$, если $T(1)$ нечетно) вычисление α приостанавливается, и лента A_1 прокручивается назад до тех пор, пока не достигается начало цепочки (α_1) . В этот момент лента A_1 содержит $\alpha_1\alpha_2 \dots \alpha_{k_1}$ для $k_1 \geq 1$.

В то время как лента A_1 прокручивается назад, на ленте A_3 записываются k_1 единиц. Вычисление α задерживается до момента $T(1)$, а в этот момент выдается α_1 . Вычисление α возобновляется до момента времени $(T(k_1 + 1) + T(1))/2 = T(1) + (T(k_1 + 1) - T(1))/2$, а этот момент может быть отмечен счетчиком $T/2$ (который также был приостановлен от момента $T(1)/2$ до $T(1)$) и с помощью единиц с ленты A_3 (которые постепенно стираются на этом этапе). На этом этапе работы машины все новые вычисленные элементы α записываются на ленту A_2 .

В момент $(T(k_1 + 1) + T(1))/2$ вычисление α и работа счетчика $T/2$ опять приостанавливаются. В этот момент лента A_2 содержит $\alpha_{k_1+1} \dots \alpha_{k_2}$ для некоторого $k_2 \geq k_1 + 1$. Лента A_2 прокручивается назад, записывая на ленте A_3 $(k_2 - k_1)$ единиц. От момента окончания прокручивания назад ленты A_2 до момента $T(k_1 + 1)$ работа машины состоит только в выдаче результатов описываемым ниже способом.

T -счетчик машины работает параллельно со всем описанным выше и сигнализирует машине, что надо печатать символы $\alpha_2, \alpha_3, \dots, \alpha_{k_1}$ в моменты $T(2), T(3), \dots, T(k_1)$ соответственно. Таким образом, к моменту $T(k_1 + 1)$ лента A_1 опять оказывается пустой.

Процесс повторяется снова: новые члены α записываются на ленте A_1 по мере их появления до момента $(T(k_2 + 1) + T(k_1 + 1))/2$, а элементы α , хранимые на ленте A_2 , печатаются на выходе в моменты времени $T(k_1 + 1), T(k_1 + 2), \dots, T(k_2)$ и т. д.

Пусть кружок обозначает композицию функций, т. е. $U \circ V(n) = U(V(n))$. Теперь мы можем сформулировать основную теорему.

Теорема 1. Пусть R и V принадлежат F_n . Тогда для всякого U , такого, что $S_{V \circ R} - S_{U \circ R} \neq \emptyset$, $S_V - S_U \neq \emptyset$.

Доказательство. Пусть α — последовательность из $S_{V \circ R} - S_{U \circ R}$, а $\mathcal{T}$ — машина, порождающая α точно за $V(R(n))$. Определим последовательность символов β , такую, что

$$\beta_k = \begin{cases} \alpha_n, & \text{если } k = R(n) \text{ для некоторого } n, \\ 0 & \text{в противном случае.} \end{cases}$$

Очевидно, что $\beta \in S_V$, так как последовательность β может быть порождена $\mathcal{T}$ в сочетании с машиной, вычисляющей $V(n)$ в реальное время и печатающей нули в каждый момент $V(n)$, где n не является значением R . Предположение $V \in F_n$ обеспечивает существование такой машины для V . Запишем в табличной форме:

β	Номер символа	Число операций
0	0	$V(0)$
0	1	$V(1)$
0	2	$V(2)$
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	⋮
0	$R(0) - 1$	$V(R(0) - 1)$
α_0	$R(0)$	$V(R(0))$
0	$R(0) + 1$	$V(R(0) + 1)$
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	⋮
0	$R(1) - 1$	$V(R(1) - 1)$
α_1	$R(1)$	$V(R(1))$
0	$R(1) + 1$	$V(R(1) + 1)$
⋮	⋮	⋮
⋮	⋮	⋮
⋮	⋮	⋮

Теперь предположим, что $\beta \in S_U$. Пусть $\mathcal{T}'$ есть машина, U -вычисляющая β . Так как $\mathcal{T}'$ печатает только $R(n)$ -е цифры β , ее можно видоизменить так, чтобы она порождала α_n за $U(R(n))$ операций. Но это противоречит первоначальному предположению о том, что $\alpha \in S_{V \circ R} - S_{U \circ R}$. Таким образом, $\beta \in S_V - S_U$.

Теорему 1 можно использовать для доказательства некоторых результатов относительно S_U - (и F_U)-иерархии. Например,

$$(1) \quad S_{n!} \subsetneq S_{(n+1)!}.$$

Доказательство. Рассмотрим $S_{n!} \overset{A}{\subset} S_{(n+1)!} \overset{B}{\subset} S_{(n+2)!}$.
 $S_{n!} \neq S_{(n+2)!}$ ввиду $\lim_{n \rightarrow \infty} \frac{n! \log n!}{(n+2)!} = 0$.

Следовательно, включение A или включение B должно быть собственным. Если B собственное, то, полагая $R(n) = n+1$, $U(n) = n!$, $V(n) = (n+1)!$ и применяя теорему 1, получаем $S_{n!} \overset{\subset}{\neq} S_{(n+1)!}$, т. е. включение A является собственным в любом случае.

$$(2) \quad S_{2^n} \overset{\subset}{\neq} S_{n2^n}, \text{ и вообще}$$

$$S_{2^N} \overset{\subset}{\neq} S_{N2^N},$$

$$\text{где } N = 2^{2^{\dots 2^n}} \Bigg\}^k.$$

Доказательство. Докажем это для класса S самого низкого порядка; другие случаи аналогичны. Рассмотрим

$$S_{2^{2^n}} \overset{A}{\subset} S_{2^n \cdot 2^{2^n}} = S_{2^{2^n+n}} \overset{B}{\subset} S_{(2^n+n)2^{(2^n+n)}}.$$

Как и выше, критерий $T \log T$ влечет $S_{2^{2^n}} \overset{\subset}{\neq} S_{(2^n+n)2^{2^n+n}}$, поскольку

$$\lim_{n \rightarrow \infty} \frac{2^{2^n} \log 2^{2^n}}{(2^n+n) 2^{(2^n+n)}} = 0.$$

Таким образом, по крайней мере одно из включений A или B является собственным. Если A собственное, то, полагая $R(n) = 2^n$, $U(n) = 2^n$, $V(n) = n \cdot 2^n$ и применяя теорему 1, получаем $S_{2^n} \overset{\subset}{\neq} S_{n2^n}$. Если B — собственное включение, то, полагая

$$R(n) = 2^n + n, \quad U(n) = 2^n, \quad V(n) = n \cdot 2^n$$

и применяя теорему 1, также получаем $S_{2^n} \overset{\subset}{\neq} S_{n2^n}$.

$$(3) \quad S_{2^{n^2}} \overset{\subset}{\neq} S_{n^2 \cdot 2^{n^2}}.$$

Доказательство. Из предыдущего имеем

$$S_{2^{(2^{2^n}-1)^2}} \overset{\subset}{\neq} S_{2^{2^n}} \overset{\subset}{\neq} S_{2^{2^n} \cdot 2^{2^{2^n}}} = S_{(2^{2^n}-1)^2 \cdot 2^{(2^{2^n}-1)^2}}.$$

В силу результатов Ямада $R(n) = 2^{2^{n-1}} = 2^{(2^n/2)}$ удовлетворяет предположению теоремы 1. Полагая $U(n) = 2^{n^2}$, а $V(n) = n^2 \cdot 2^{n^2}$, мы докажем собственное включение.

В каждом из этих примеров (и многих других подобных) приведено собственное включение $S_U \subsetneq S_V$, где не удовлетворяется критерий Хенни — Стирнза, а $\lim \frac{U \log U}{V} = c > 0$ или даже равен ∞ .

Таким образом, каждый пример дает уточнение иерархии. Приложения теоремы 1 используют следующее общее свойство S_U -иерархии.

Следствие. Для любых $U, T \in F_n$ $S_U \subsetneq S_{U \circ T^k}$ влечет $S_U \subsetneq S_{U \circ T}$, где $T^k(n)$ обозначает k -кратную композицию T (т. е. если некоторое число внутренних композиций расширяет класс, то и одна композиция расширяет его).

Доказательство. По предположению одно из включений в последовательности $S_U \subset S_{U \circ T} \subset S_{U \circ T \circ T} \subset \dots \subset S_{U \circ T^k}$ должно быть собственным.

Пусть j будет такое число, что $S_{U \circ T^j} \subsetneq S_{U \circ T^{j+1}}$. Полагая $U(n) = U(n)$, $V(n) = U(T(n))$, $R(n) = T^j(n)$ и применяя теорему 1, получаем утверждение следствия.

Следующие предположения, будь они верны для $T, U, R \in F_n$, помогли бы уяснить свойства перевода в иерархии.

- (A) $S_T \subsetneq S_U$ влечет для всех R $S_{T \circ R} \subsetneq S_{U \circ R}$ (см. теорему 2);
- (B) $S_T \subsetneq S_U$ влечет для всех R $S_{R \circ T} \subsetneq S_{R \circ U}$ (обращение (B) ложно);
- (C) $S_T \subsetneq S_U$ тогда и только тогда, когда

$$S_{R(n) \cdot T(n)} \subsetneq S_{R(n) \cdot U(n)}.$$

Теорема 1 позволяет «переводить» некоторые собственные включения между классами высокого порядка (например, $S_{U \circ R}$ и $S_{V \circ R}$) в собственные включения между классами низшего порядка (например, S_U, S_V). В следующем разделе мы дадим пример, когда происходит обратный перевод.

СУЩЕСТВЕННЫЕ ФУНКЦИИ

Допустимая функция T называется *существенной* для $S_U \subsetneq S_V$ тогда и только тогда, когда существует двоичная последовательность α с $\alpha_k = 0$, если только k не равно $T(n)$ для некоторого n и $\alpha \in S_V - S_U$. Основой для этого определения служит следующий результат.

Теорема 2. Если $U, T \in F_n$ и T является существенной функцией для $S_U \subsetneq S_V$, то $S_{U \circ T} \subsetneq S_{V \circ T}$.

Доказательство. Пусть $\alpha \in S_V - S_U$ есть такая последовательность, что $\alpha_k = 0$, если неверно, что $k = T(n)$ для некоторого n . Тогда $\alpha_T = \alpha_{T(0)}\alpha_{T(1)}\alpha_{T(2)} \dots \in S_{V \circ T} - S_{U \circ T}$. Если $\mathcal{T}$ — машина Тьюринга, которая V -вычисляет α , то $\alpha_{T(n)}$ порождается за $V(T(n))$ операций машины $\mathcal{T}$. Поскольку $T \in F_n$, то все выходы, кроме $T(0), T(1), \dots, T(n)$ -го выходов $\mathcal{T}$, могут быть исключены за $V(T(n))$ операций. В результате машина будет $V(T(n))$ -вычислять α_T . Если вдобавок α_T порождается машиной $\mathcal{T}'$, которая вычисляет α_T точно за время $U(T(n))$, то, выдавая на выход 0 на $U(k)$ -й операции, когда k не является значением T , можно изменить $\mathcal{T}'$ так, чтобы она порождала α_k за $U(k)$ операций, что противоречит условию теоремы.

Справедливо также обращение теоремы 2.

Теорема 3. Из $V, T \in F_n; S_{U(T(n))} \subsetneq S_{V(T(n))}$ следует, что T является существенной функцией для $S_U \subsetneq S_V$.

Доказательство. В доказательстве теоремы 1 β является последовательностью, которая делает T существенной функцией.

Если $R(T(n))$ — существенная функция для $S_U \subsetneq S_V$, то R существенна для $S_U \subsetneq S_V$, так как любая последовательность α , с помощью которой устанавливается существенность $R \circ T$, такую же роль играет в отношении R .

Теорема 4. $U, T \in F_n; \lim_{n \rightarrow \infty} \frac{T \log T}{U} = 0$ влечет существенность R для $S_T \subsetneq S_U$ для любого $R \in F_n$.

Доказательство. Рассмотрим любое $R \in F_n$. Тогда $S_{T \circ R} \subsetneq S_{U \circ R}$ потому, что

$$\lim_{n \rightarrow \infty} \frac{T(R(n)) \log T(R(n))}{U(R(n))} = \lim_{m \rightarrow \infty} \frac{T(m) \log T(m)}{U(m)} = 0.$$

По теореме 3 R существенна для $S_T \subsetneq S_U$.

F_U -ИЕРАРХИЯ

Наряду с S_U -иерархией обычно рассматривают тесно с ней связанную (напомним, что $T \in F_U$ тогда и только тогда, когда $\varphi(T) \in S_U$) F_U -иерархию. Классы F_U обладают интересными

функциональными свойствами. Можно сказать, что у F_U есть некоторый статус «естественного» разделения вычислимых монотонных функций. Например, если $U \in F_n$, то класс F_U замкнут относительно композиции. Этот факт является следствием из следующей теоремы «перевода».

Теорема 5. Если $U \in F_n$, то $T \in F_U$ тогда и только тогда, когда $U \circ T \in F_n$.

Доказательство. Предположим, что $T \in F_U$. Поскольку $U \in F_n$, по лемме машину $\mathcal{T}$ для $\varphi(T)$ можно выбрать так, чтобы она печатала $\varphi(T)_n$ в своей $U(n)$ -й операции. Машина, которая печатает выходы машины $\mathcal{T}$, когда их печатает машина $\mathcal{T}$, и 0 в остальные моменты, порождает $\varphi(U \circ T)$ в реальное время. Действительно, пусть последовательность α — выход этой машины; тогда $\alpha_k = 0$, если только не имеют места $k = U(n)$ и $n = T(i)$ для некоторого i , но это является определением $\varphi(U \circ T)$. Обратно, предположим, что $\mathcal{T}'$ порождает $\varphi(U \circ T)$ в реальное время. Пусть $\mathcal{T}''$ порождает в реальное время $\varphi(U)$. Если печатать выход $\mathcal{T}'$ только тогда, когда текущий выход $\mathcal{T}''$ есть 1, получим машину, которая U -вычисляет $\varphi(T)$; n -й выход этой новой машины соответствует $U(n)$ -му символу $\varphi(U \circ T)$, который будет 1 тогда и только тогда, когда $n = T(i)$ для некоторого i , но это является определением $\varphi(T)$.

Следствие. Если $V \in F_n$, то для всех $U, T \in F_V$ имеем $U \circ T \in F_V$.

Доказательство. Предположим, что $U \circ T \notin F_V$, тогда по теореме 5 $V \circ U \circ T \notin F_n$. Но $U \in F_V$ влечет (по той же теореме), что $V \circ U \in F_n$, следовательно, теорему 5 можно применить снова для вывода $T \notin F_{V \circ U} \supset F_V$, что противоречит условию (в действительности для доказательства в условии достаточно принять, что $T \in F_{V \circ U}$).

Вышесказанное дает возможность предположить, что можно обобщить все результаты Ямады, т. е. доказать, что для $V \in F_n$ класс F_V замкнут относительно композиции, сложения, умножения и экспоненцирования, а это, конечно, желательные свойства для функциональной иерархии.

ЗАКЛЮЧЕНИЕ

Приложение метода перевода будет зависеть в итоге от доказательства теорем мультипликативного, экспоненциального и других типов перевода, аналогичных теоремам 1 и 5, где фигурирует композиционный перевод. Некоторую надежду на это дает тот факт, что класс F_n наряду с композицией замкнут относительно других операций. Теорема 1 подсказывает возмож-

ное направление исследований S_U - и F_U -иерархий. Можно привести некоторые соображения для установления свойств замкнутости классов F_U и S_U . Заслуживает внимания также рассмотрение подклассов классов сложности, связанных с машинами Тьюринга с фиксированным числом k лент.

ЛИТЕРАТУРА

1. Hartmanis J., Stearns R. E., On the Computational Complexity of Algorithms, *Trans. Am. Math. Soc.*, 117 (1965), 285—306. (Русский перевод: Хартманис Дж., Стернс Р. Е., О вычислительной сложности алгоритмов, Кибернетический сборник (новая серия), вып. 4, «Мир», М., 1967, стр. 57—85.)
2. Hennie F., Stearns R. E., Two-tape simulation of multi-tape Turing machines, *J. ACM*, 13, № 4 (1966), 533—546. (См. стр. 194—212 настоящего сборника.)
3. Yamada H., Real-time computation and recursive functions not real-time computable, *IRE Trans. El. Comp.*, EC-11 (1962), 753—760. (См. стр. 139—155 настоящего сборника.)

ВЫЧИСЛЕНИЯ НА ОДНОЛЕНТОЧНОЙ МАШИНЕ ТЬЮРИНГА С ЗАПИСЬЮ НА ЛЕНТЕ¹⁾

Ф. К. Хенчи

1. ВВЕДЕНИЕ

В этой работе мы будем рассматривать машины Тьюринга, которые на каждом данном шаге своих вычислений могут производить следующие действия: а) менять текущий символ на ленте, обозреваемый считывающей головкой; б) сдвигать каждую из своих лент на один квадрат влево или вправо; в) менять свое внутреннее состояние; г) останавливаться. Будем считать, что каждый шаг для своего выполнения требует ровно одну единицу времени. Для того чтобы исследовать полное время, требующееся для вычисления данной функции, удобно различать on-line и off-line тьюринговы вычисления. Эти два типа вычисления различаются только способом, которым входные данные поступают в машину и выходные данные порождаются ею. В on-line вычислениях (или в *вычислениях на машине Тьюринга со входом*) входные данные поступают в машину по одному символу за единицу времени на специальное входное устройство. Каждый входной символ требует от машины выдачи через специальное выходное устройство соответствующего выходного символа. Вообще говоря, машина не может сделать это в непосредственно следующий момент времени, так как она должна затратить некоторое время, вычисляя требуемый выходной символ. В то же время новый символ не может быть подан на входное устройство, пока машина вычисляет выходной символ, который соответствует предыдущему входному. Таким образом, каждый выходной символ есть функция исключительно предшествующего входного символа. В работах [1—3] используется эта модель машины Тьюринга.

В off-line вычислениях (или в *вычислениях на машине Тьюринга с записью на ленте*) входные символы записываются на одной из лент машины перед началом вычислений. Результаты вычислений получаются тогда и только тогда, когда машина останавливается, и их можно считать либо словом, записанным на одной из лент, либо внутренним состоянием, в котором находится машина после остановки.

¹⁾ Hennie F. C., One-tape, off-line Turing machine computations, *Information and Control*, 8, № 6 (1965), 553—578.

В этой работе мы будем рассматривать только вычисления с записью на ленте, в частности такие вычисления этого типа, которые имеют только два возможных выхода. Эти выходы, представляемые символами 0 и 1, могут считаться отождествленными с заключительными состояниями машины.

Все ленты, используемые при вычислениях с записью на ленте, предполагаются односторонне бесконечными, ограниченными слева и не ограниченными справа. Это не вызывает уменьшения возможностей или скорости вычисления и удобно для последующего анализа.

Перед началом вычисления входное слово должно быть записано на левом конце одной из лент машины, предназначенной для этой цели. Входное слово имеет конечную длину, и остальные квадраты ленты пусты.

В начальный момент считывающая головка обзревает крайний левый непустой квадрат и машина находится в начальном состоянии. Затем машина совершает последовательность операций, определенную ее внутренней структурой. Если машина останавливается в состоянии, которое обозначает выход 0, то будем говорить, что она **отвергает** входное слово; если она останавливается в состоянии, которое обозначает выход 1, то будем говорить, что она **допускает** входное слово. Если машина всегда останавливается через конечное время независимо от того, какое входное слово представлено ей, то будем говорить, что она **распознает** множество входных слов, которые порождают на выходе 1.

Можно считать, что такие машины классифицируют входные слова на те, которые машина допускает, и на те, которые она отвергает. В большинстве случаев мы будем иметь дело с машинами, которые всегда останавливаются независимо от представленного им конкретного входного слова. Число основных операций, которое требуется машине, чтобы допустить или отвергнуть входное слово, будет называться **временем вычисления** для этого входного слова. Если $T(n)$ — такая функция, что время вычисления для каждой входной последовательности длины n меньше либо равно $T(n)$, тогда говорим, что вычисление на машине **ограничено** $T(n)$. Аналогично, множество входных слов, которое распознает машина, назовем **распознаваемым за время** $T(n)$.

Эта работа исследует поведение одноленточных машин Тьюринга с записью на ленте с двумя целями: первая цель — развить метод для описания вычислений на таких машинах и вторая — применить этот метод к проблеме нахождения хороших нижних границ времени, требующегося для распознавания различных множеств слов.

В разд. 2 дается понятие «следа» и изучаются свойства этого понятия, которые делают его важным аналитическим средством для одноленточных вычислений с записью на ленте.

В разд. 3 определяются нижние границы для времени вычисления. В частности, в разд. 3, А описывается множество слов, время распознавания которого должно быть более чем $C_1 n^2 / \log Q$, однако не превосходит $C_2 n^2 / \log Q$, где C_1 и C_2 — соответствующие константы, Q — число внутренних состояний машины Тьюринга, распознающей это множество.

В разд. 3, В рассматривается более общее время вычисления вида Kn^p , где K и p — действительные константы. Для времени вычисления этого вида может быть установлен следующий результат: каковы бы ни были числа p и q , где $1 \leq q < p \leq 2$, существует множество слов, время распознавания которого пропорционально n^p , однако его нельзя распознать за время пропорциональное n^q , предполагая, что распознавание происходит на одноленточной машине Тьюринга с записью на ленте.

2. СЛЕДЫ

А. Определения. Напомним, что вычисления с записью на ленте начинаются с соответствующего входного слова, записанного на одну из лент машины. Ограничим наше рассмотрение машинами, имеющими только одну ленту. Эта лента должна быть использована не только для записи входных слов, но также и для любых промежуточных записей в процессе вычисления. Как было уже указано, удобно предполагать, что лента бесконечна вправо, но слева ограничена. Входное слово занимает конечный сегмент на крайнем левом конце ленты, а оставшаяся часть ленты пуста. Перед началом вычислений считывающая головка устанавливается на крайний левый квадрат входного слова. Можно потребовать, чтобы этот квадрат был помечен специальным символом, чтобы считывающая головка не сошла с ленты. В вычислениях, рассматриваемых в этой работе, такой специальный конечный маркер не является необходимым, и мы не будем его рассматривать. Включение конечного маркера не потребует существенных изменений в дальнейшем обсуждении.

В этой работе термины *лента* и *слово* мы используем для обозначения двух различных понятий. Термин *лента*, если не оговорено противное, означает бесконечную цепь квадратов, в которые можно записывать некоторые символы. Если необходимо сослаться на часть ленты, то мы будем использовать термин *сегмент ленты*. В частности, если t_a обозначает конечный сегмент на левом конце ленты и если t_b означает оставшийся бесконечный сегмент той же ленты, то символ $t_a t_b$ можно использо-

вать для обозначения всей бесконечной ленты. Термин «слово», с другой стороны, означает последовательность символов, записанную на ленте в данный момент. Такая последовательность должна всегда иметь конечную длину. В большинстве случаев будет необходимо ссылаться только на слово, записанное на ленте машины Тьюринга перед началом вычислений. Тогда мы будем говорить о *начальной ленте* и *входном слове*, которое она содержит. Когда описываются вычисления на машинах Тьюринга, удобно считать, что считывающая головка движется по ленте, а лента остается неподвижной.

Зигзагообразная линия на рис. 1 показывает примерный путь считывающей головки в течение всего процесса вычисления. Мы

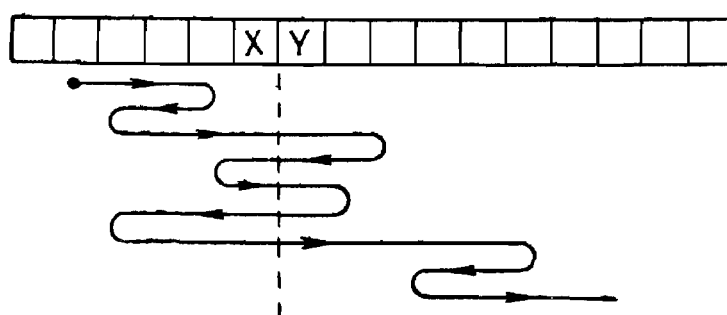


Рис. 1. Путь вычислений.

будем обычно рассматривать этот путь как пройденный по самому входному слову и игнорировать способ, которым меняется слово в течение всего процесса вычисления. С другой стороны, нас будут интересовать внутренние состояния, в которые попадает машина на каждом шаге своих вычислений, и мы будем обозначать путь этими состояниями.

Рассмотрим теперь два соседних квадрата ленты, обозначенных, например, X и Y на рис. 1, и отметим точки в вычислении, в которых головка пересекает границу между двумя квадратами. Так как каждое вычисление начинается с того момента, когда считывающая головка находится на левом конце ленты, то каждое нечетное пересечение между двумя данными соседними квадратами должно быть пересечением слева направо; аналогично каждое четное пересечение должно быть справа налево. Пусть $S(i)$ — состояние машины в момент i -го пересечения границы между двумя квадратами. Тогда последовательность $S = S(1), S(2), \dots, S(i), \dots$ назовем *следом*¹⁾, порожденным данной машиной Тьюринга на границе между двумя квадратами.

¹⁾ След соответствует понятию схемы, использованной Рабином [3]. [Понятие следа независимо ввел и исследовал Б. А. Трахтенброт [4*, 5*]. Он же и Я. М. Барздинь доказали утверждения, эквивалентные теореме 1 и следствию 1 [4*, 6*]. — Перев. и ред.]

Содержательно след описывает способ, которым машина передает информацию посредством своих внутренних состояний, переходя с одной стороны границы на другую. Определение следов, связанных со всеми границами, которые машина достигает в процессе вычисления, эквивалентно определению каждого основного шага в этом вычислении. В частности, общее время вычисления равно общему числу пересечений, которое делает машина. Если число пересечений назвать *длиной следа*, то общее время вычисления равно сумме длин всех следов. Очевидно, машина остановится при данной начальной ленте тогда и только тогда, когда она порождает конечное число непустых следов, каждый из которых конечной длины.

В. Основные свойства. Хотя мы прежде всего интересуемся машинами, которые должны всегда в конце концов останавливаться независимо от начальной ленты, мы временно ослабим эти ограничения. Тогда машина с записью на ленте может реагировать на начальную ленту трояко: может остановиться и допустить ленту, может остановиться и отвергнуть ленту или может продолжать вычисления бесконечно. Две начальные ленты называются *эквивалентными* относительно данной машины тогда и только тогда, когда они либо обе допускаются машиной, либо обе отвергаются, либо на каждой из них машина никогда не закончит вычисления.

Теорема 1. Пусть $t_a t_b$ — начальная лента, состоящая из конечного сегмента t_a и следующего за ним справа бесконечного сегмента ленты t_b . Заметим, что t_a не обязательно совпадает с частью ленты, которая изначально непуста.

Аналогично пусть $t_c t_d$ — начальная лента, состоящая из конечного сегмента t_c и следующего за ним справа бесконечного сегмента t_d .

Предположим, что $t_a t_b$ и $t_c t_d$ эквивалентны относительно данной машины Тьюринга M . Пусть C_1 и C_2 — следы, порожденные машиной на границе между t_a и t_b и t_c и t_d соответственно. Если C_1 равен C_2 , то справедливы следующие утверждения.

- (А) Начальные ленты $t_a t_b$ и $t_c t_d$ эквивалентны относительно данной машины M .
- (В) Вычисления, производимые над $t_a t_b$ и $t_c t_d$, порождают следы на соответствующих границах частей лент t_a , которые должны быть тождественными.
- (С) Вычисления, производимые над $t_a t_d$ и $t_c t_d$, порождают следы на соответствующих границах частей лент t_d , которые должны быть тождественными.

Другими словами, часть t_c ленты $t_c t_d$ можно заменить на t_a без изменения конечного результата на ленте, если след между t_a и t_d не меняется. Рис. 2, а, б показывают типичные пути гипотетической машины Тьюринга на лентах $t_a t_b$ и $t_c t_d$. Если след, порожденный на границе между t_a и t_b , равен следу, порожденному на границе между t_c и t_d , как показано, тогда вычисление, произведенное на ленте $t_a t_d$, должно идти путем, показанным на рис. 2, в. Это изображение делает теорему интуитивно почти очевидной, и поэтому мы не хотим давать ее строгого доказательства. Приведем полезное следствие, почти непосредственно вытекающее из теоремы 1.

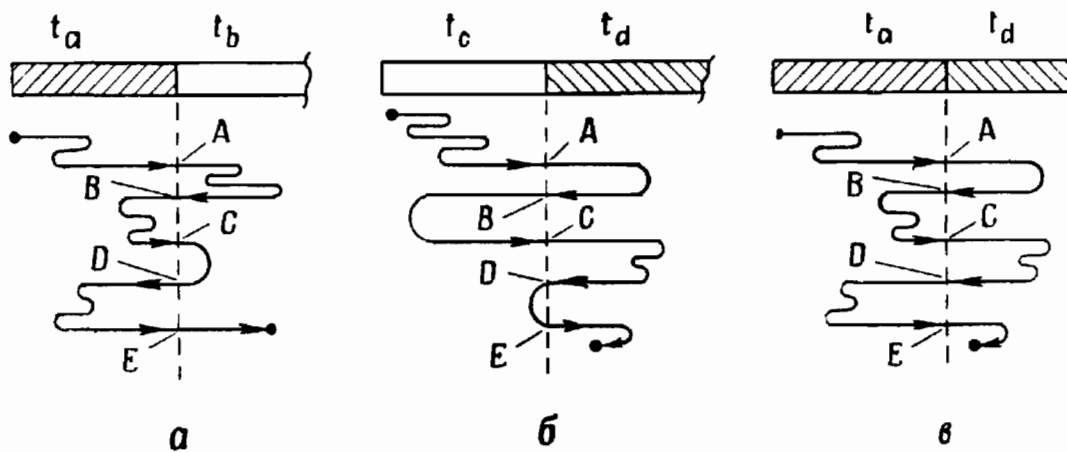


Рис. 2. Иллюстрация теоремы 1.

Следствие 1. Если для начальной ленты $t_1 t_2 t_3$ след, порожденный данной машиной M на границе между t_1 и t_2 , равен следу, порожденному на границе между t_2 и t_3 , то

- (А) ленты $t_1 t_2 t_3$ и $t_1 t_3$ эквивалентны относительно машины M ;
- (В) ленты $t_1 t_2 t_3$ и $t_1 t_2 t_2 t_3$ эквивалентны относительно машины M .

Другими словами, если одинаковые следы встречаются по обе стороны одного и того же сегмента ленты, то этот сегмент может быть удален и соседние сегменты соединены без изменения конечного результата вычислений либо дополнительный экземпляр (или экземпляры) данного сегмента может быть вставлен внутрь без изменения конечного результата вычисления.

Доказательство. Часть (А). Пусть t_a , t_b , t_c и t_d представляют соответственно сегменты ленты t_1 , $t_2 t_3$, $t_1 t_2$ и t_3 . Тогда $t_a t_b$ и $t_c t_d$ — тождественные ленты (а именно $t_1 t_2 t_3$) и, конечно, будут эквивалентны относительно машины M . Тогда по теореме 1 $t_a t_b$ и $t_a t_d$ должны быть эквивалентны. Но $t_a t_b = t_1 t_2 t_3$, а $t_a t_d = t_1 t_3$.

Часть (В). Пусть $t_a = t_1 t_2$, $t_b = t_3$, $t_c = t_1$ и $t_d = t_2 t_3$, и снова воспользуемся теоремой.

Хотя теорема 1 и следствие 1 достаточны для установления временных границ, обсуждаемых в следующем разделе, мы про-

ведем более детальное исследование идеи, затронутой в теореме. Этим мы можем добиться лучшего понимания соотношения между понятием следа и более привычным понятием внутреннего состояния в приложении к автоматам. Теорема 1 описывает ситуацию, в которой можно сегмент t_c , являющийся левым концом ленты $t_c t_d$, заменить другим сегментом t_a без изменений результата вычислений. В частности, это может быть сделано, если существует вторая лента $t_a t_b$, которая эквивалентна $t_c t_d$ и для которой машина порождает тот же самый след в правом конце сегмента t_a , что и в правом конце сегмента t_c . Такие сегменты, как t_a , t_c , встречающиеся на левом конце ленты, назовем левоконечными сегментами. Рассмотрим теперь отношение между двумя левоконечными сегментами t_1 и t_2 , выполняющееся тогда и только тогда, когда t_2 можно *всегда* подставить вместо t_1 независимо от полного слова, в котором последний встречается. Очевидно, такая подстановка возможна только тогда, когда каждый след, встречающийся на конце t_1 , можно встретить на конце t_2 . При этом необходимо точное понимание процедуры, определяющей, может ли данный конечный след встретиться на правом конце данного левоконечного сегмента. Это невозможно сделать, рассматривая все ленты, содержащие данный левоконечный сегмент, так как, вообще говоря, необходимо было бы проделать бесконечное число вычислений, чтобы убедиться, что данный след не встречается на конце данного сегмента. Вместо этого для любой данной машины Тьюринга, конечного левоконечного сегмента t и конечного следа $C = S(1), \dots, S(\lambda)$ мы можем провести следующий эксперимент.

1) Начнем эксперимент, помещая машину в начальное состояние и заставляя ее обозревать крайний левый квадрат t .

2) Если машина, покидая сегмент t в i -й момент времени ($i < \lambda$), находится в состоянии $S(i)$, то переведем ее в состояние $S(i+1)$ и отправим назад в самый правый квадрат t .

3) Если машина останавливается внутри t , или впадает в некоторый цикл внутри t , или покидает t таким образом, что правило 2 не выполняется, то эксперимент заканчивается.

Очевидно, такой эксперимент закончится через конечное число шагов. Если в конце эксперимента след на правом конце t есть в точности C , то говорим, что сегмент t имеет след C на своем правом конце. Если длина C нечетна, то в конце эксперимента машина находится вне сегмента t . В этом случае C будет называться *переходным* следом для t . Если длина C четна, то в конце эксперимента машина находится внутри сегмента t . Если машина останавливается и выдает на выходе 1, то след C называем допускающим следом для t , в противном случае C называется недопускающим следом для t .

Теперь предположим, что каждый конечный след, принадлежащий данному левоконечному сегменту ленты t_1 , принадлежит некоторому другому сегменту t_2 . Далее предположим, что каждый непереходный след, допускающий для t_1 , допускающий и для t_2 . Тогда рассмотрим произвольную ленту, в которой t_1 встречается как левоконечный сегмент и на которой машина производит конечные вычисления.

Из теоремы 1 следует, что t_2 может быть заменен на t_1 без изменения результата работы машины над первоначальной лентой или следов, встречающихся в части ленты справа от t_1 .

Наконец, предположим, что а) каждый след, принадлежащий t_1 , принадлежит и t_2 , и обратно; б) каждый непереходный след, допускающий для t_1 , допускающий и для t_2 , и обратно. Тогда левоконечные сегменты t_1 и t_2 можно взаимно заменять без изменения результата вычислений.

Классифицируя левоконечные сегменты ленты в соответствии со следами, которые они имеют, и в соответствии с теми из них, которые являются допускающими, можно показать, что эта классификация порождает классификацию входных слов, соответствующую некоторому автомату.

С. Простые приложения. Теперь поучительно рассмотреть специальный случай, в котором следы, порожденные машиной, не превосходят некоторой длины, скажем K . Другими словами, машина никогда не проводит в одном и том же квадрате своей ленты более чем K единиц времени. Легко видеть, что если машина никогда не бывает в любом данном квадрате своей ленты более чем K раз, то существует лишь конечное число различных следов, которые могут встретиться в любом квадрате. Далее, существует только конечное число различных подмножеств этих следов. Если известно подмножество следов, которые могут порождаться машиной в данном квадрате ленты, тогда можно определить подмножество следов, которое может порождаться машиной в следующем квадрате справа. Следовательно, машина может точно с таким же успехом побывать в каждом квадрате точно один раз, передвигаясь по ленте слева направо в соответствии с подмножествами следов. Эта нить рассуждений приводит нас к следующей теореме.

Теорема 2. *Если длина каждого следа в каждом вычислении данной одноленточной машины Тьюринга с записью на ленте не превосходит K , то существует автомат, распознающий в точности то же самое множество входных слов, что и данная машина Тьюринга¹⁾.*

¹⁾ Утверждение теоремы 2 доказано Б. А. Трахтенбротом [4*]; в [7*] установлены близкие друг к другу верхняя и нижняя оценки для числа состояний автомата, существование которого утверждается в теореме 2.—Прим. перев. и ред.

Доказательство. Доказательство заключается в показе того, что множество слов, которое распознает данная машина Тьюринга, можно представить как объединение некоторого числа классов конечного право-инвариантного отношения эквивалентности.

Для любой данной машины Тьюринга и для любого данного конечного числа K существует не более конечного числа различных следов, длина которых не превосходит K . Далее, число подмножеств этих следов конечно.

Каждому конечному левоконечному сегменту ленты ставим в соответствие след длины K или меньшей длины, который он имеет на своем правом конце и который является либо переходным, либо допускающим, либо недопускающим; в соответствии с этим происходит разбиение таких сегментов на классы.

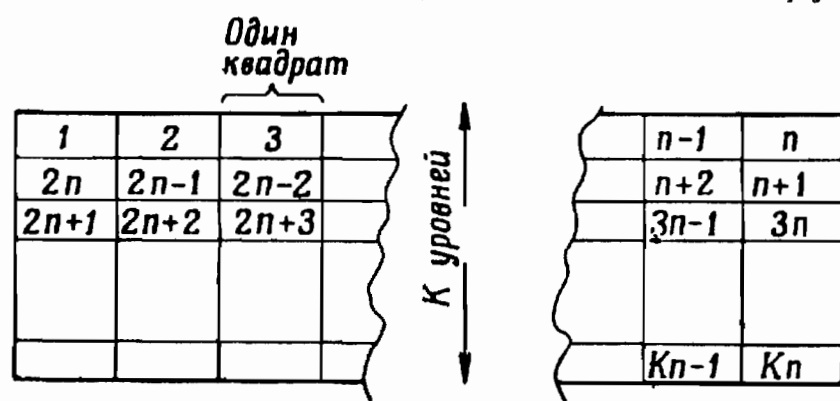
Теперь предположим, что два конечных левоконечных сегмента t_1 и t_2 принадлежат к одному и тому же классу, как только что описано. Тогда рассмотрим любые левоконечные сегменты ленты вида $t_1 t_x$, где t_x — конечный сегмент со следом длины $\leq K$. Очевидно, любой след, который принадлежит $t_1 t_x$, будет также принадлежать и $t_2 t_x$, и обратно, так как в эксперименте, описанном в разд. 2, В, t_1 можно заменить на t_2 без изменения следов в части t_x . Далее, любой непереходный след, который является допускающим для $t_1 t_x$, будет допускающим также для $t_2 t_x$ и любой непереходный след, не допускающий для $t_1 t_x$, будет не допускающим также и для $t_2 t_x$.

Таким образом, $t_1 t_x$ и $t_2 t_x$ принадлежат одному и тому же классу левоконечных сегментов ленты. Так как это верно для любого t_x , то вышеописанная классификация есть конечное право-инвариантное отношение эквивалентности.

Мы должны показать теперь, что множество допустимых входных слов есть объединение некоторого числа классов эквивалентности. Пусть слово p допустимо данной машиной Тьюринга. Если q — другое слово, то, в случае когда оно из того же класса эквивалентности, q должно быть допустимым машиной, так как мы видели, что элементы данного класса эквивалентности можно заменять друг на друга без изменения результата работы машины над лентой, содержащей данный левоконечный сегмент. Следовательно, если один элемент некоторого класса эквивалентности допустим, то каждый элемент этого класса должен быть допустим. Аналогично если некоторый элемент класса недопустим, тогда в этом классе ни один элемент не допустим. Отсюда следует, что множество допустимых слов образует объединение некоторого числа классов эквивалентности. Поэтому получаем, что множество допустимых слов можно распознать автоматом, что и требовалось доказать.

Теорема 2 устанавливает, что если некоторая машина Тьюринга тратит на посещение любого своего квадрата ленты не более чем некоторое фиксированное число единиц времени, то она может быть заменена автоматом.

Далее предположим, что максимальное время, которое затрачивает машина на посещение каждого квадрата, не ограничено, однако ограничено в среднем время на квадрат входного слова. Другими словами, предположим, что полное время вычисления на машине не превосходит Kn , где n — длина входного слова. В таком случае машина может порождать на некоторых границах очень длинные следы, тогда как на других границах



Р и с. 3. Лента с несколькими уровнями.

достаточно короткие. Но, несмотря на это, можно показать, что любую машину Тьюринга такого типа можно также заменить автоматом.

Прежде чем доказывать этот факт, заметим, что мы можем ограничить наше внимание машинами Тьюринга, которые никогда не покидают входных частей своих лент, т. е. частей лент, на которых написаны исходные слова. Предположим, что время вычисления машины M не превосходит Kn . Такая машина, конечно, не может посетить более чем $(K-1)n$ пустых квадратов ленты. Построим теперь новую машину M' , ленты которой разделены на K уровней, как показано на рис. 3. Символы, которые могут встречаться теперь в квадратах новых лент, представляют упорядоченные комбинации из K символов первоначальных лент, по одному символу на каждом уровне клетки ленты. Верхний уровень новой ленты используется для записи входного слова, в то время как оставшиеся $K-1$ уровней рассчитаны на $(K-1)n$ квадратов пустой ленты, которые может использовать машина M . Машина M' начинает работу только с символов, которые встречаются в верхнем уровне ее ленты, и работает точно так же, как и M в начале своих вычислений. Однако, если M пытается выйти за пределы входной последовательности в пустую часть ленты, M' возвращается обратно и работает в обратном направлении во втором уровне своей ленты. Если M

двигается за n -й квадрат пустой ленты, M' продолжает работать в правом направлении в третьем уровне своей ленты. В результате пустая часть ленты машины M складывается зигзагообразным образом во входную часть ленты. Хотя мы не даем деталей, ясно, что всегда возможно построить таким образом машину M' , которая распознает то же самое множество входных слов, что и данная машина M .

Теорема 3. Если время работы одноленточной машины Тьюринга ограничено функцией $T(n) = Kn$, где K — константа, тогда существует автомат, который распознает в точности то же самое множество входных слов, что и данная машина Тьюринга¹⁾.

Доказательство. Доказательство состоит в том, что показывается, что если полное время вычислений ограничено числом Kn , тогда время пребывания в любом квадрате ленты должно быть ограничено числом $2KQ^K + K$, где Q — число внутренних состояний. Тогда утверждение теоремы будет немедленно следовать из теоремы 2.

Предположим, что машина Тьюринга имеет Q внутренних состояний и устроена таким образом, что никогда не выходит за пределы входного слова. Затем предположим, что в процессе вычисления на некотором входном слове машина порождает след, длина которого больше чем $2KQ^K + K$. В частности, пусть n_0 — наименьшая длина входного слова, на котором машина порождает такой «длинный» след.

Выберем теперь некоторое входное слово длины n_0 , на котором машина порождает след такой длины, и рассмотрим вычисление данной машины по преобразованию этого слова. Пусть s есть число следов в этом вычислении, длина которых меньше чем K . Имея в виду, что общее время вычисления T равно сумме длин всех следов в вычислении, мы имеем

$$Kn_0 \geq T(n_0) > 2Q^K K + K + (n_0 - 1 - s)K,$$

отсюда

$$s > 2Q^K,$$

однако число различных следов длины, меньшей чем K , есть

$$\sum_{i=1}^{K-1} Q^i \leq Q^K - 1,$$

так как данная машина имеет только Q состояний.

Следовательно, среди более чем $2Q^K$ границ, в которых след является «коротким», должны существовать по крайней мере три

¹⁾ В [4*] доказано, что теорема 3 остается верной для $T(n)$, таких, что $\lim T(n)/(n \log n) = 0$. — Прим. перев. и ред.

границы, в которых след одинаковый. Назовем эти границы b_1, b_2 и b_3 , и пусть b_0 — граница, в которой имеется «длинный» след.

По крайней мере две из границ b_1, b_2 и b_3 должны находиться по одну и ту же сторону от b_0 . Предположим, что b_1 и b_2 расположены так, как показано на рис. 4.

Теперь построим новое входное слово, удаляя часть первоначального слова между b_1 и b_2 и соединяя концы кусков друг с другом. В соответствии со следствием 1 следы, которые порождает данная машина, производя свои вычисления на новой лен-

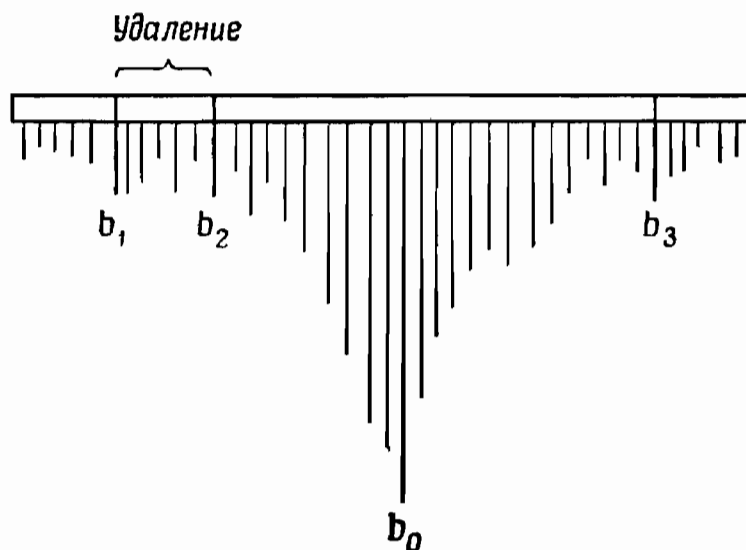


Рис. 4. Иллюстрация теоремы 3.

те, должны быть тождественны тем следам, которые порождаются на соответствующих границах старой ленты. В частности, так как b_0 находится вне части старой ленты между b_1 и b_2 , то вычисления на новой ленте порождают след, длина которого больше чем $2KQ^K + K$. Длина нового слова меньше чем n_0 . Однако n_0 определялось как наименьшая длина входного слова, на котором машина порождает «длинный» след. Следовательно, предположение, что на некотором конечном входном слове машина порождает след длины больше чем $2KQ^K + K$, ведет к противоречию. Следовательно, нет ни одного следа, порожденного машиной, длины, большей, чем $2KQ^K + K$, независимо от входного слова, представленного машине. В соответствии с теоремой 2 должен существовать автомат, распознающий то же самое множество входных слов, что и данная машина Тьюринга. Ч. т. д.

Теореме 3 легко дать следующую интерпретацию.

Предположим, что некоторое множество входных слов можно распознать одноленточной машиной Тьюринга, время вычисления которой для входного слова прямо пропорционально длине соответствующего входного слова.

Тогда это же множество слов можно распознать одноленточной машиной Тьюринга, время вычисления которой для входного слова в точности равно длине этого слова. Эта последняя

машина Тьюринга просто перемещается слева направо по своей ленте подобно автомату. Таким образом, вычислительные возможности одноленточных машин Тьюринга не возрастают, если разрешить времени вычисления изменяться от n до Kn .

Интересно сравнить этот результат с соответствующими результатами относительно других способов вычислений. Используя методы Хартманиса и Стирнза [2], можно показать, что если некоторое множество слов можно распознать на одноленточной машине Тьюринга за время $KT(n)$, тогда его можно распознать за время $T(n)$, предполагая $T(n) > n^2$. Другими словами, если время вычисления достаточно большое (больше чем n^2), то увеличение времени вычисления на постоянный множитель не увеличивает вычислительных возможностей одноленточной машины Тьюринга. Кажется разумным предположение, что это верно для любой функции $T(n)$, однако понятие следа, по-видимому, неудобно в установлении этого результата для $T(n) > n$, а методы Хартманиса и Стирнза, по-видимому, неудобны для $T(n) \leq n^2$. В настоящее время неизвестно, возрастают ли вычислительные возможности машины Тьюринга при увеличении времени вычисления от $T(n)$ до $KT(n)$, если $n^2 \geq T(n) > n$. Сходные вопросы можно поставить для двухленточных машин с записью на ленте. Снова при увеличении времени вычисления от $T(n)$ до $KT(n)$ вычислительные возможности машины не возрастают при $T(n) > n^2$. Однако при увеличении времени вычисления с n до Kn , очевидно, возрастают и вычислительные возможности двухленточных машин. Где в точности между n и n^2 использование постоянного множителя не меняет вычислительных возможностей машины, еще не известно, однако очевидно, что прибавление второй ленты изменяет эти возможности для машин с записью на ленте.

Дальнейшие различия между одно- и двухленточными машинами излагаются в следующем разделе.

3. ВРЕМЕННЫЕ ГРАНИЦЫ ДЛЯ ОДНОЛЕНТОЧНЫХ ВЫЧИСЛЕНИЙ

А. Распознавание множества S . Мы рассмотрим теперь проблему определения верхних и нижних границ времени, необходимого для распознавания некоторых множеств входных слов на одноленточных машинах с записью на ленте. Во-первых, приведем пример множества слов, для которого можно показать, что минимально необходимое время вычисления $T(n)$ заключено в границах

$$C_1 n^2 / \log Q \leq T(n) \leq C_2 n^2 / \log Q,$$

где n — длина входного слова и Q — число внутренних состоя-

машина Тьюринга просто перемещается слева направо по своей ленте подобно автомату. Таким образом, вычислительные возможности одноленточных машин Тьюринга не возрастают, если разрешить времени вычисления изменяться от n до Kn .

Интересно сравнить этот результат с соответствующими результатами относительно других способов вычислений. Используя методы Хартманиса и Стирнза [2], можно показать, что если некоторое множество слов можно распознать на одноленточной машине Тьюринга за время $KT(n)$, тогда его можно распознать за время $T(n)$, предполагая $T(n) > n^2$. Другими словами, если время вычисления достаточно большое (больше чем n^2), то увеличение времени вычисления на постоянный множитель не увеличивает вычислительных возможностей одноленточной машины Тьюринга. Кажется разумным предположение, что это верно для любой функции $T(n)$, однако понятие следа, по-видимому, неудобно в установлении этого результата для $T(n) > n$, а методы Хартманиса и Стирнза, по-видимому, неудобны для $T(n) \leq n^2$. В настоящее время неизвестно, возрастают ли вычислительные возможности машины Тьюринга при увеличении времени вычисления от $T(n)$ до $KT(n)$, если $n^2 \geq T(n) > n$. Сходные вопросы можно поставить для двухленточных машин с записью на ленте. Снова при увеличении времени вычисления от $T(n)$ до $KT(n)$ вычислительные возможности машины не возрастают при $T(n) > n^2$. Однако при увеличении времени вычисления с n до Kn , очевидно, возрастают и вычислительные возможности двухленточных машин. Где в точности между n и n^2 использование постоянного множителя не меняет вычислительных возможностей машины, еще не известно, однако очевидно, что прибавление второй ленты изменяет эти возможности для машин с записью на ленте.

Дальнейшие различия между одно- и двухленточными машинами излагаются в следующем разделе.

3. ВРЕМЕННЫЕ ГРАНИЦЫ ДЛЯ ОДНОЛЕНТОЧНЫХ ВЫЧИСЛЕНИЙ

А. Распознавание множества S . Мы рассмотрим теперь проблему определения верхних и нижних границ времени, необходимого для распознавания некоторых множеств входных слов на одноленточных машинах с записью на ленте. Во-первых, приведем пример множества слов, для которого можно показать, что минимально необходимое время вычисления $T(n)$ заключено в границах

$$C_1 n^2 / \log Q \leq T(n) \leq C_2 n^2 / \log Q,$$

где n — длина входного слова и Q — число внутренних состоя-

дения сравнивает два или более символов из первого и третьего блоков. В частности, если машина при каждом прохождении сравнивает k символов, то число прохождений, а следовательно, и общее время вычисления уменьшаются приблизительно в k раз. Но такая схема вычисления требует, конечно, много состояний. Не вдаваясь в детали, можно заметить, что если при каждом прохождении машина сравнивает k символов, то ее можно устроить с не более чем $(k+1)2^{k+1}$ состояниями. Можно показать, что соответствующее время вычисления при этом не превосходит

$$\frac{2n^2}{\log Q} + 4n \quad (\text{для } Q \geq 8),$$

где Q — число внутренних состояний и логарифм берется по основанию 2. Таким образом, множество S можно распознать на одноленточной машине Тьюринга с записью на ленте, время вычисления которой для больших n растет прямо пропорционально n^2 и обратно пропорционально $\log Q$, равному числу бит внутренней памяти.

Попытаемся теперь найти хорошую нижнюю границу для наилучшего возможного времени вычисления в терминах числа состояний и длины входного слова.

Предположим, что нам дана одноленточная машина Тьюринга с записью на ленте, имеющая Q внутренних состояний и распознающая множество S .

В последующем обсуждении мы будем рассматривать только допустимые входные слова некоторой произвольно выбранной длины n , где n кратно 3. Существует, конечно, ровно $2^{n/3}$ таких слов. Для каждого из таких слов машина, проводя вычисления, порождает конечные следы в каждой из $(n-1)$ границ внутри слова. Посмотрим, какие из этих следов должны быть различными и какие могут быть одинаковыми. Так как мы рассматриваем только допустимые слова, то из теоремы 1 и следствия 1 следует, что два следа могут быть одинаковыми в том и только в том случае, если результат соединения сегмента ленты, находящегося слева от одного из следов, с сегментом ленты, находящимся справа от другого из следов, дает допустимое слово. Легко показать, что это может произойти, только если два следа встречаются на соответствующих местах внутри первого (или третьего) блока двух различных вычислений и если слова, находящиеся слева (справа) от этих двух следов, тождественны. В таком случае, разрезая две начальные ленты в рассматриваемых границах и меняя их левые (правые) концы, мы не изменяем всего слова в целом. При любом другом расположении двух следов, разрезая начальную ленту или ленты в некоторых границах и соединяя концы, получаем по крайней мере одно

недопустимое слово. Следовательно, следы, встречающиеся на этих границах, должны быть различными.

В частности, все следы, встречающиеся на границах внутри центрального блока допустимых слов длины n , должны быть различными. Если мы рассмотрим две границы внутри центрального блока одного и того же слова, то, удаляя сегмент между границами, получаем в результате допустимое слово без достаточного числа двоек. Если мы рассмотрим две границы внутри центрального блока различных слов, то, меняя концы, получим в результате слово, в котором не совпадают первый и третий блоки. Если бы следы на этих двух границах были тождественны, то это противоречило бы следствию 1 или теореме 1. Мы можем теперь использовать этот факт при определении нижней границы для максимального времени, которое должна затратить данная машина при работе над некоторым допустимым словом длины n . Так как использованные доводы не принимают во внимание следы, встречающиеся в первом и третьем блоках, то нельзя утверждать, что мы получим наилучшую возможную границу. Однако рассмотрение всех следов требует довольно длинного доказательства, в то время как время вычисления при нашем подходе возрастает в два раза.

Пусть s означает число различных следов, длина которых не превосходит $\lambda = n/(3 \log Q) - 1$, где Q — число состояний данной машины, а логарифм берется по основанию 2. Заметим, что число различных следов длины 1 есть Q , число следов длины два есть Q^2 и т. д. Следовательно,

$$s = \sum_{i=1}^{\lambda} Q^i < Q^{\lambda+1} = Q^{n/(3 \log Q)} = 2^{n/3} \quad (\text{для } Q \geq 2).$$

Однако число допустимых слов длины n в точности равно $2^{n/3}$, которое больше чем s . Так как один и тот же след не может встречаться в центральной части при двух различных вычислениях, то мы видим, что не достаточно одних только «коротких» следов, чтобы они встречались в центральной части при каждом вычислении.

Отсюда существует такое вычисление, преобразующее допустимое слово длины n , для которого длина каждого следа в центральной части превосходит λ .

Количество времени, которое должна затратить данная машина при этом вычислении, равно сумме длин следов, встречающихся в этом вычислении. Соответственно количество времени, затраченное машиной в центральном блоке слова, есть по крайней мере $(n/3)(\lambda+1)$, так как в этом блоке существует $n/3$ точек и в каждой точке длина следа $\lambda+1$ или более. Отсюда общее время вычисления по крайней мере $n^2/(9 \log Q)$.

Если учесть все следы в вычислении, преобразующем допустимое входное слово длины n , то эту нижнюю границу можно удвоить. Таким образом, если n кратно 3, должно существовать некоторое допустимое слово длины n , над которым машина работает по крайней мере $2n^2/(9 \log Q)$ единиц времени.

Итак, если множество S распознается одноленточной машиной Тьюринга с Q состояниями, то соответствующее время вычисления должно превосходить $2n^2/(9 \log Q)$, где n кратно 3, однако может не превосходить $2n^2/\log Q + 4n$. В частности, для $Q = 8$ время вычисления должно превосходить $2n^2/27$, в то время как мы знаем, что можно построить машину с восемью состояниями, время работы которой не превосходит $12n^2/27 + n$. Таким образом, для больших n оптимальное время вычисления на машине с восемью состояниями определяется в пределах коэффициента 6.

Поучительно постараться определить, хотя бы неформально, почему верхняя и нижняя границы различаются множителем 6.

Для того чтобы это сделать, нужно соответствующим образом понять задачу одноленточной машины по передаче данных, когда она проходит через различные границы своей ленты. Уменьшение времени вычисления достигается тем, что каждый индивидуальный член следа должен нести больше информации. Конечно, фиксируя число внутренних состояний, мы фиксируем также максимальный «объем» информации, который может быть передан каждым отдельным членом следа. Если достигается наименьшее возможное время вычисления, то каждый член следа используется на полную мощность. Отсюда для любого выбора внутреннего состояния и целого числа i должен найтись след, в котором данное состояние встречается на i -м месте.

Возвращаясь к вышеописанному методу для распознавания множества S , мы видим, что некоторые члены следов не используют полностью своих возможностей по передаче информации.

В частности, всякий раз когда машина двигается справа налево, она, завершив одно прохождение, просто возвращается в начальную точку следующего, и полезной информации почти не передается. Таким образом, все члены следов с четными номерами почти полностью затрачиваются впустую.

Ввиду уже одного этого факта мы должны ожидать, что требуемое время вычисления при настоящем методе будет почти в два раза больше, чем данное нижней оценкой.

Производительность реализации уменьшается и другими «внутрисимвольными» ограничениями, которые машина накладывает на свои следы. Эти ограничения возникают главным образом из-за необходимости машине знать, в каком блоке входного слова находится ее считывающая головка. Таким образом,

обязанность машины вести «домашнее хозяйство» отрывает ее от основной работы по передаче информации о входном слове, и это значительно уменьшает вычислительную производительность машины. Способом борьбы с этим обстоятельством должно быть изменение метода, использованного при распознавании множества S . В случае модификации вычислительной схемы и значительного расширения внешнего алфавита становится возможным сконструировать машину с восемью состояниями, время вычисления которой приблизительно равно удвоенной данной нижней границе. Таким образом, в этом частном примере можно получить относительно высокую производительность передачи информации; первоначально данная нижняя граница будет относительно хорошей.

Нужно помнить, однако, что доводы, использованные для получения нижней границы времени вычисления, не учитывают того обстоятельства, какие встречающиеся следы может породить одноленточная машина Тьюринга. Так как модель машины Тьюринга сама налагает ограничения на следы, которые могут встречаться в данном вычислении, то нельзя ожидать, что аргументы, аналогичные аргументам этого раздела, приведут к хорошей нижней оценке для всех проблем распознавания¹⁾.

Проблема распознавания множества S интересна по нескольким причинам.

Во-первых, она иллюстрирует приложение понятия следа к проблеме определения минимального времени вычисления. Во-вторых, она дает специфический пример вычислительной проблемы, для которой можно получить достаточно близкие верхние и нижние границы времени. В-третьих, она дает некоторое представление о соотношении между следами, данной вычислительной схемой и «производительностью» этой схемы. Наконец, она дает информацию о соотношении скоростей одно- и двуленточных машин, которая рассматривается ниже.

Рассмотрим проблему построения двуленточной машины Тьюринга с записью на ленте, которая распознает множество S . Такая машина начинает свои вычисления, если входное слово записано на одной из ее лент, называемой **входной лентой**. Вторая лента, называемая **внешней лентой**, перед началом вычислений полностью пуста. Простой способ выполнения желаемого вычисления состоит из следующих трех шагов.

¹⁾ Результат, аналогичный изложенному выше, содержится в [6*] в более сильной форме: там построен пример такого множества, что распознавание принадлежности к нему для почти всех слов длины n требует времени порядка n^2 . Соотношению между числом состояний машины и временем ее работы на словах длины $\leq n$, о котором говорит автор, посвящена работа [5*]. — Прим. перев. и ред.

Шаг 1. Машина продвигает обе свои ленты справа налево, переписывая первый блок входного слова на внешнюю ленту. Общее время, требующееся для этого шага, равно длине первого блока.

Шаг 2. Машина продолжает двигать входную ленту налево, но теперь внешняя лента продвигается слева направо, так что машина проходит назад над словом, которое только что было записано на этой ленте.

Таким образом, машина сравнивает длину второго блока (входной ленты) с длиной первого блока (внешней ленты). Если длины не одинаковы, вычисление немедленно прекращается. Если длины одинаковы, машина обзревает первый символ третьего блока входной ленты и первый символ первого блока внешней ленты. В любом случае время, необходимое для этого шага, не превосходит числа символов во втором блоке входного слова.

Шаг 3. Машина продвигает обе ленты справа налево, сравнивая слово, которое находится в третьем блоке (входной ленты), со словом, которое находится в первом блоке (внешней ленты). Время, необходимое для этого шага, не превосходит числа символов в третьем блоке входного слова. Таким образом, если n — длина всего входного слова, общее время, необходимое для вычисления, не более n ; время вычисления ограничено, конечно, функцией $T(n) = n$.

Итак, для приведенного примера время наилучшего вычисления, которого можно достичь в случае одноленточной машины с записью на ленте, пропорционально квадрату времени наилучшего вычисления, которого можно достичь в случае двуленточной машины. С другой стороны, из работы Хартманиса и Стирнза [2] известно, что если данная двуленточная машина завершает свои вычисления за $T_2(n)$ единиц времени, то должна существовать одноленточная машина, которая завершает свои вычисления за время $C[T_1(n)]^2$, где C — константа. Другими словами, переходя от двуленточной машины к одноленточной, никогда не нужно требовать более чем квадрата времени вычисления. Однако теперь мы имеем пример, в котором квадрат необходим. Таким образом, «закон квадрата» Хартманиса — Стирнза не может быть, вообще говоря, улучшен.

В. Порядок скорости роста для других множеств. Множество S есть пример проблемы распознавания, для которой необходимое время вычисления растет пропорционально n^2 и которую можно разрешить за время, пропорциональное n^2 . Таким образом, мы можем считать n^2 «порядок скорости роста»,

связанным с распознаванием на одноленточной машине Тьюринга. В этом разделе мы вкратце рассмотрим проблемы распознавания с другими порядками скорости роста, в частности порядками скорости роста вида n^p , где p — рациональное число между 1 и 2.

В качестве первого примера мы хотим исследовать множество входных слов, время распознавания которого растет как $n^{3/2}$. Это множество аналогично множеству S и отличается от него относительными длинами трех блоков. Новое множество, обозначенное как множество R , состоит в точности из тех слов, которые удовлетворяют каждому из следующих условий.

1. Каждое слово состоит в точности из трех блоков: блок из 0 и 1, следующий за ним блок из 2 и следующий за ним блок из 0 и 1.

2. Пусть n обозначает длину всего слова, а x — длину первого блока. Тогда

(А) x есть степень 2,

(В) n равно x^2 .

3. Слова, находящиеся в первом и третьем блоках, тождественны. (Отсюда следует, что первый и третий блоки имеют одинаковую длину.)

Таким образом, слово 011122222220111 принадлежит множеству R , в то время как 011122220111 не принадлежит R , так как его длина не равна квадрату длины первого блока. Множество R удобно распознавать машиной, вычисления которой состоят из двух последовательных стадий. На первой стадии машина определяет, удовлетворяются ли условия 1 и 2; на второй стадии определяется, удовлетворяется ли условие 3. Условие 1 можно очень просто проверить, проходя ленту один раз, на что требуется только n единиц времени. Условие 2(А) эквивалентно требованию, что $\log x$ есть целое число, в то время как условие 2(В) эквивалентно требованию, что $\log x = (1/2) \log n$. Следова-

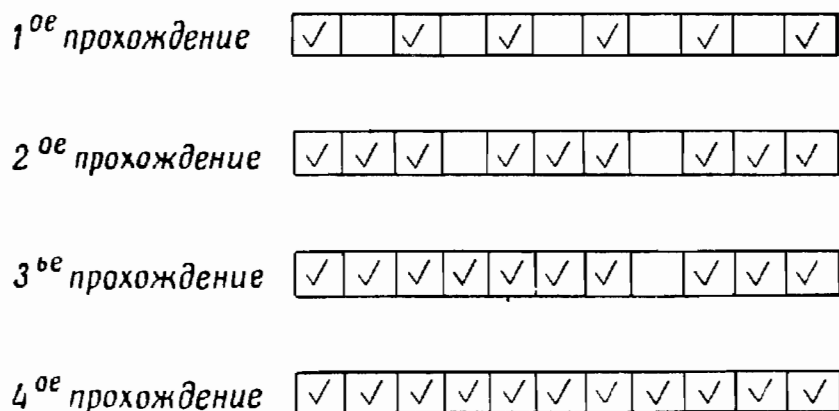


Рис. 5. Метод для определения логарифма,

тельно, машина, построенная для проверки условия 2, вычисляет логарифмы x и n .

Машина, вычисляющая целую часть логарифма по основанию 2 от длины блока ленты, проделявает последовательность сквозных проходов блока. При каждом проходе она отмечает специальным символом первый, третий, пятый и т. д. ранее не отмеченные квадраты, как показано на рис. 5. Число проходов, необходимых для того, чтобы отметить все квадраты, на единицу больше, чем целая часть от логарифма длины этого блока. Далее, длина блока будет степенью 2 (и ее логарифм будет целым) тогда и только тогда, когда правый крайний квадрат в блоке отмечается последним.

Машина, определяющая, удовлетворяется ли условие 2, должна выполнять две маркерные операции: одну над целым словом, другую над первым его блоком. Эти две операции можно различить, используя различные маркерные символы. Более удобно сконструировать машину так, что она делает два прохода по целой ленте для каждого прохода первого блока. Тогда условие 2 удовлетворяется тогда и только тогда, когда две маркерные операции завершаются на одном и том же проходе и в каждом случае последний символ отмечается на последнем проходе. Каждое прохождение требует самое большее $2n$ единиц времени, и максимальное число необходимых проходов пропорционально логарифму n . Таким образом, общее время, необходимое для проверки условий 1 и 2, не более чем $\alpha_1 n \log n$, где α_1 — константа. Наконец, условие 3 можно проверить способом, аналогичным использованному при распознавании множества S . Машина опять делает последовательность проходов ленты и при каждом проходе сравнивает один символ первого блока с одним символом из третьего блока. Время, требующееся на каждое прохождение, меньше чем $2n$, в то время как число проходов $x = n^{1/2}$. Таким образом, время, необходимое для проверки условия 3, меньше чем $2n^{3/2}$, и время, необходимое для полного вычисления, меньше чем $2n^{3/2} + \alpha_1 n \log n$, которое меньше чем $\alpha_2 n^{3/2}$. Другими словами, множество R можно распознать за время, растущее так же, как и $n^{3/2}$. Что порядок скорости роста $n^{3/2}$ необходим для распознавания R , можно установить с помощью приемов, аналогичных использованным в предыдущем разделе. Если мы не касаемся конкретного коэффициента пропорциональности, то достаточно рассматривать только следы, которые порождаются в центральном блоке допустимых слов длины n . Заметим, что существует в точности $2^{n^{1/2}}$ различных допустимых слов длины n (предполагая n степенью 4). С другой стороны,

число различных следов, длина которых не превосходит $\mu = n^{1/2}/\log Q - 1$, есть $\sum_{i=1}^{\mu} Q^i < Q^{\mu+1} = 2^{n^{1/2}}$ (для $Q \geq 2$).

Таким образом, коротких следов меньше, чем вычислений, и некоторое вычисление должно иметь в центральном блоке только следы, длина которых больше чем μ . Так как число границ в центральном блоке по крайней мере $n/2$, то общее время, требующееся для этого вычисления, по крайней мере

$$T(n) = (\mu + 1) \frac{n}{2} = \frac{n^{3/2}}{2 \log Q}.$$

Другими словами, для любого фиксированного числа состояний время вычисления, необходимое для распознавания множества R , должно расти пропорционально $n^{3/2}$.

Определение множества R можно теперь модифицировать так, чтобы описанные множества имели различные порядки скорости роста в интервале от n до n^2 .

Следующие условия определяют множества слов, порядок скорости роста которых $n^{1+q/r}$, где q, r — целые и $q < r$:

1. Каждое слово состоит в точности из трех блоков: блок из 0 и 1, следующий за ним блок из 2 и следующий за ним блок из 0 и 1.
2. Пусть n обозначает длину всего слова, и пусть x обозначает длину первого блока. Тогда
 - (А) x есть степень 2^q .
 - (В) $x = n^{q/r}$.
3. Слова, находящиеся в первом и третьем блоках, равны.

Машину, распознающую такое множество, можно сконструировать почти таким же способом, что и машину, распознающую множество R . Как и прежде, условие 1 совсем легко проверяется. Условия 2(А) и 2(В) одновременно эквивалентны следующим требованиям:

$$(1/q) \log x = (1/r) \log n = \text{целое}.$$

Следовательно, в то время как машина вычисляет логарифмы x и n , она выполняет последовательность из r проходов целого слова, затем q проходов первого блока, затем снова r проходов целого слова и т. д. Условие 2 удовлетворяется тогда и только тогда, когда

- (а) маркерная операция для целого слова заканчивается после некоторого целого числа k последовательностей из r проходов;
- (б) маркерная операция для первого блока заканчивается после того же целого числа k последовательностей из q проходов;

(с) на последнем шаге каждой маркерной операции отмечается последний квадрат соответствующего сегмента ленты, т. е. $\log x$ и $\log n$ — целые числа.

Условие (3) можно проверить обычным образом. Общее время вычисления не более чем

$$\alpha_4 n \log n + \alpha_5 n (n^{q/r}) \leq \alpha_6 n^{1+q/r}.$$

То, что порядок скорости роста $n^{1+q/r}$ необходим для распознавания данного множества, можно показать способом, аналогичным тому, который использовался для множества R .

С. Общие результаты. Результаты, полученные в двух предыдущих разделах, можно объединить в следующую теорему.

Теорема 4. Для любого рационального числа p в интервале $1 < p \leq 2$ существует множество входных слов S_p , такое, что

- (а) S_p можно распознать за время вычисления, меньшее чем $C_1(p, Q)n^p$;
- (б) S_p нельзя распознать за время вычисления, меньшее чем $C_2(p, Q)n^p$,

где C_1 и C_2 — функции только от p и числа внутренних состояний, используемых при вычислении¹⁾.

Следствие 4. Если p и q — два действительных числа в интервале $1 \leq q < p \leq 2$, тогда существует множество входных слов, которое может быть распознано за время вычисления, пропорциональное n^p , и не может быть распознано за время вычисления, пропорциональное n^q .

Доказательство. Выберем, во-первых, такое рациональное r , что $q < r < p$. Тогда в соответствии с теоремой 4 должно существовать множество слов, которое можно распознать за время $C_1(r, Q)n^r$, однако нельзя распознать за время $C_2(r, Q)n^r$. Так как $C_1(r, Q)n^r < C_1(r, Q)n^p$, то это множество, конечно, можно распознать за время, пропорциональное n^p . Теперь предположим, что существует некоторая машина с Q состояниями, которая распознает то же самое множество за время Cn^q . Однако для достаточно большого n $Cn^q < C_2(r, Q)n^r$, что противоречит предположению. Таким образом, это множество нельзя распознать за время вычисления, пропорциональное n^q , ч. т. д.

Как ранее подчеркивалось, кажется разумным предположение, что если некоторое множество входных слов можно распознать на одноленточной машине Тьюринга с записью на ленте за время $KT(n)$, тогда его можно также распознать за время $T(n)$ при условии $T(n) \geq n$. Другими словами, кажется разумным предположить, что всякое вычисление можно ускорить в

¹⁾ Аналогичный результат для машин со входом получен в [7*]. — Прим. перев и ред.

постоянное число раз, используя дополнительные внутренние состояния. Однако, согласно теореме 4, это есть наилучшее ускорение, которое можно получить для произвольной проблемы распознавания, по крайней мере в интервале времени вычисления от n до n^2 .

Кроме того, допустимое время не нуждается в очень сильном возрастании для того, чтобы обеспечить машинам способность распознавать новые множества. Согласно следствию 4, увеличение любой временной границы n^p ($1 \leq p < 2$) всего в n^ε раз ($\varepsilon > 0$) расширяет вычислительные способности одноленточных машин. Итак, иерархия классов сложности (Хартманис и Стирнз, [2]) всюду плотна по крайней мере в интервале от n до n^2 .

4. ЗАКЛЮЧЕНИЕ

Следы дают удобное средство описания способа, которым в одноленточной машине Тьюринга передается информация из одной части ленты в другую. Их роль можно сравнить с ролью внутренних состояний автомата.

Наличие данного состояния в автомате определяет один класс из конечного числа классов, который содержит входное слово, полученное машиной. Наличие данного следа при вычислении на машине Тьюринга определяет два класса: один, который содержит часть начальной ленты слева от следа, и другой, который содержит часть начальной ленты, лежащей справа от следа. В каждом случае определяется один класс из бесконечного числа классов.

Таким образом, внутреннее состояние и след дают некоторую информацию о входном слове, которое поступило в соответствующую машину. В то время как внутреннее состояние дает информацию только о слове, которое встречается по одну сторону от границы (прошлое), след дает информацию о словах, которые встречаются по обе стороны от границы. Далее, состояние автомата может обозначать только конечное число предшествующих классов, в то время как след машины Тьюринга может обозначать бесконечное число пар классов слов. Когда мы работаем с автоматом, часто можно сделать вывод, что два особым образом выбранных входных слова должны привести машину в различные внутренние состояния для того, чтобы машина в дальнейшем вела себя надлежащим образом. Почти таким же образом часто возможно сделать вывод, что следы, встречающиеся на особым образом выбранных границах в одном или более вычислениях, должны быть различны, для того чтобы машина Тьюринга, их порождающая, вела себя соответствующим образом для всевозможных входных слов. Знание того, что состояния, получающиеся в результате некоторых вход-

ных слов, должны быть различны, дает нам возможность установить границу числа состояний автомата. Знание того, что некоторые следы должны быть различны, также дает нам возможность установить нижнюю границу времени, требующегося для вычислений на одноленточной машине Тьюринга. К сожалению, такие границы обычно не так просто получить, как аналогичные границы для числа состояний автомата.

Временные границы, полученные рассмотрением следов, не обязательно должны быть близкими. Существуют две причины для этого. Во-первых, не всегда целесообразно принимать во внимание все различные следы, которые существуют среди следов, порождаемых данной машиной Тьюринга. Так, может быть, необходимо не учитывать часть работы машины при получении нижней границы времени вычисления. Во-вторых, доводы, использованные для получения границ, предполагают, что машина использует свои следы для передачи информации с максимальной эффективностью. Практически это не всегда возможно либо потому, что машина должна вернуться, «идя с пустыми руками» из некоторой отдаленной части своей ленты, либо потому, что она должна затратить значительное время на организацию или на кодирование данных для приведения их к виду, в котором они могут быть использованы в другой части ленты.

Это последняя проблема, проблема кодировки данных в удобной форме, мешает использованию очень полезной идеи следа для проблем, требующих времени вычисления больше чем n^2 . Так, рассмотрим экстремальный случай, в котором все следы во всех вычислениях на всевозможных входных словах длины, меньшей либо равной n , должны быть различны. Даже в этом случае доводы, использованные в предыдущем разделе, требуют среднего времени вычисления, растущего так же, как n^2 . Это максимальная скорость роста при идеальных условиях для сообщения каждому квадрату ленты полной информации о символах, первоначально записанных в каждом другом квадрате ленты. Это, конечно, хитрость, так как просто сообщения этих данных недостаточно. Информация должна быть передана в такой форме, чтобы она могла быть правильно интерпретирована машиной в точке, в которой делается вывод относительно допустимости или недопустимости слова. Так как этот вывод должен быть сделан сведением к устройству с конечным числом состояний (чья входы есть состояния машины Тьюринга, а внутренние состояния есть символы ленты), то размер каждого отдельного следа ограничен. В таких случаях длина следа может быть много больше длины, определяемой рассмотрением только передачи данных, и вообще время вычисления может быть намного больше, чем определяемое доводами в разд. 3. Хотя идею следа

можно распространить на двуленточные машины с записью на ленте, но она, по-видимому, неудобна для получения временных границ для таких машин. Возможно, наиболее уместный прием для рассмотрения вычислений на двуленточной машине — считать, что считывающая головка двигается в плоскости; координаты точки в этой плоскости определяются положением головки на каждой из двух лент. Тогда возможно определить след как последовательность внутренних состояний, в которых машина входит в данный квадрат на плоскости и выходит из него. К сожалению, кажется, нет прямых аналогов теоремы 1 или следствия 1, которые применимы к случаю плоскости, и, следовательно, весь метод нахождения нижних границ не проходит. Несмотря на ограниченную область приложения, понятие следа может быть использовано для получения определенных результатов о некоторых вычислениях на машинах Тьюринга. Разумеется, класс одноленточных машин Тьюринга с записью на ленте — один из немногих классов машины Тьюринга, для которых могут быть сделаны такие сильные утверждения, как теорема 4. Наиболее важным результатом является то, что этот класс машин Тьюринга является единственным, для которого годится любая идея, отражающая возможности и способности концепции конечного числа внутренних состояний.

ЛИТЕРАТУРА¹⁾

1. Yamada H., Real-time computation and recursive functions not real-time computable, *IRE Trans. Electron. Computers*, EC-11 (1962), 753—760. (См. настоящий сборник стр. 139—155.)
2. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Am. Math. Soc.*, 117, № 5 (1965), 285—306. (Русский перевод: Кибернетический сборник (новая серия), вып. 4, 1967.)
3. Rabin M. O., Real-time computation, *Israel J. Math.*, 1 (1963), 203—211. (См. настоящий сборник стр. 156—167.)
- 4*. Трахтенброт Б. А., Тьюринговы вычисления с логарифмическим замедлением, *Алгебра и логика, Семинар*, 3, № 4 (1964), 33—48.
- 5*. Трахтенброт Б. А., О нормированных сигнализирующих для тьюринговых вычислений, *Алгебра и логика, Семинар*, 5, № 6 (1966), 61—70.
- 6*. Барздин Я. М., Сложность распознавания симметрии на машинах Тьюринга, *Проблемы кибернетики*, вып. 15, 1965, 245—248.
- 7*. Фрейвальд Р. В., О порядке роста точных временных сигнализирующих для тьюринговых вычислений, *Алгебра и логика, Семинар*, 5, № 5 (1966), 85—94.
- 8*. Брейтбарт Ю. Я., Козмиди В. А., О двух подклассах машин Тьюринга, сводимых к конечным автоматам, *ДАН СССР*, 187, № 1 (1969), 9—10.

¹⁾ За период, прошедший со времени написания этой работы, получено довольно много результатов, примыкающих к затронутой автором тематике. Наряду статей даются ссылки в подстрочных примечаниях (они дополнительно внесены в список литературы и отмечены звездочками), некоторые работы упомянуты в предисловии и в добавлении редактора к первому разделу настоящего сборника. — *Прим. ред.*

ВЫЧИСЛЕНИЯ НА МАШИНАХ ТЬЮРИНГА СО ВХОДОМ ¹⁾

Ф. К. Хенни

1. ВВЕДЕНИЕ

Одной из мер сложности функции является время, необходимое для вычисления этой функции на машине Тьюринга. Так как конкретная модель машины Тьюринга может существенно повлиять на время вычисления, то часто различают вычисления «с записью на ленте» и «со входом». Коротко говоря, при вычислении с записью на ленте все (конечное) слово входных символов должно быть написано на одной из машинных лент до начала вычисления, в то время как при вычислении со входом символы входного слова поступают в машину последовательно. Некоторые аспекты машин с записью на ленте обсуждались в предыдущей работе [1]; настоящая же работа касается лишь машин со входом.

Машина со входом начинает вычисление на пустых лентах и получает входные символы по одному за каждый шаг до специального заключительного символа. В ответ на каждый входной символ машина должна выработать соответствующий выходной символ на специальной выходной ленте. Между получением входного символа и выдачей соответствующего выходного символа машина должна выполнить некоторое число вычислений. Однако $(i + 1)$ -й входной символ должен поступать в машину тогда и только тогда, когда машина выработала i -й выходной символ. Таким образом, машина со входом может мыслиться как реализующая преобразование входных последовательностей в выходные последовательности, в которых i -й символ выходной последовательности зависит только лишь от первых i символов входной последовательности. Если выходной алфавит машины есть $\{0, 1\}$, то эту машину можно рассматривать как распознающую множество входных последовательностей, для которых последний символ в соответствующей выходной последовательности есть 1.

На каждом шаге вычисления машина со входом способна изменить символы, появляющиеся под ее читающими головками,

¹⁾ Hennie F. C., On-line Turing machine computations, *IEEE Trans. on Electronic Computers*, EC-15, № 1 (1966), 35—44.

сдвинуть каждую из ее лент на один квадрат в любом направлении и переменить внутреннее состояние. Общее число шагов, которые машина должна совершить, чтобы выдать выходные символы, связано с данной входной последовательностью и называется временем вычисления для этой входной последовательности. Если $T(n)$ — функция, такая, что время вычисления для каждой входной последовательности длины n меньше или равно $T(n)$, то говорят, что машина работает с временной границей $T(n)$. Входно-выходное преобразование, реализуемое машиной, будет называться «вычислимым с границей (или за время) $T(n)$ »; множество последовательностей, распознаваемых машиной, будет называться «распознаваемым с границей (или за время) $T(n)$ ».

Эта работа касается определения нижних границ времени вычисления некоторых входно-выходных преобразований и взаимосвязи между временем вычисления и размерностью лент машины Тьюринга. В разд. 2 рассматриваются одномерные ленты и описывается входно-выходное преобразование, для которого можно найти хорошую нижнюю границу времени вычисления. Метод, использованный при установлении этой границы, применяется в разд. 3, где показано, что увеличение размерности ленты от 1 до 2 иногда может значительно уменьшить время, требуемое на выполнение преобразования. Этот результат обобщается в разд. 4, где показано, что для многомерных лент «закон квадрата» Хартманиса — Стирнза [2, 3] не может быть значительно улучшен и что имеется существенная разница между размерностью и количеством лент машины Тьюринга.

2. ОДНОМЕРНЫЕ ЛЕНТЫ

В этом разделе исследуется вопрос о времени вычисления, необходимом для выполнения некоторого входно-выходного преобразования на машине Тьюринга с входом (on-line), которая использует одномерные ленты. Это преобразование может быть описано следующим образом. Машина Тьюринга снабжается сначала последовательностью входов, представляющих информацию, записанную на ленте (лентах) машины. Эта информация кодируется в форме блоков из нулей и единиц равной длины. После того как указанное число информационных блоков прочитано, машина переходит ко второй последовательности входов, также состоящей из блоков равной длины из нулей и единиц. Когда каждый из этих новых блоков прочитан, машина должна решить, был или нет такой же блок в первой части входной последовательности, а на выходе поставить 1 или 0 соответственно.

ВХОД 10020012000200020112100210020002111201121102100201
ВЫХОД 00000000000000000000000000000000000000000001000000100

При исследовании поведения машин, которые реализуют это конкретное входно-выходное преобразование, мы займемся сначала входными последовательностями, которые состоят более чем из 2^k блоков некоторой общей длины k , отделенными единственной двойкой. Первые 2^k блоков в такой последовательности называются базой последовательности, а оставшиеся блоки — хвостом последовательности. Выходной символ, выработанный в ответ на двойку, которая заканчивает блок в хвосте последовательности, будет рассматриваться как выход, связанный с этим блоком. Наконец, мы обозначим множество входных последовательностей, для которых соответствующая выходная последовательность заканчивается символом 1, через A . Тогда, например, последовательность 012002012102002112012 есть член множества A . Ее база состоит из четырех блоков 012002012102, а ее хвост — из трех блоков 002112012. Выходы, связанные с тремя хвостовыми блоками, суть 1, 0 и 1 в порядке их появления.

В соответствии с обычной терминологией машина, которая выполняет входно-выходное преобразование, описанное выше, является одной из машин, которая распознает множество A . Нетрудно себе представить, как можно построить такую машину. Когда ей подан первый блок из 0 и 1 входной последовательности, машина отмечает длину этого блока на одной из своих лент. Этот отмеченный сегмент служит потом эталоном, с которым могут сравниваться длины последующих блоков. Когда читаются блоки базовой части входной последовательности, машина сравнивает их длины, записывает их на вторую ленту и образует соответствующую последовательность выходов нулей. Конечно, машина должна определить, когда все базовые блоки прочитаны; для этого она должна вычислить значение 2^k (k определяется длиной первого блока) и подсчитать число воспринятых блоков.

Когда считываются нули и единицы хвостового блока, машина записывает их на третью ленту и образует соответствующую последовательность выходов из нулей. Как только блок заканчивается двойкой, машина начинает сравнивать только что записанный хвостовой блок с каждым из ранее записанных базовых блоков. Если найден блок, равный ему, то машина дает на выход 1, в противном случае на выход подается 0. Каждый хвостовой блок обрабатывается аналогично.

Рассмотрим теперь время, которое нужно машине, чтобы выработать выходы, связанные с данной входной последовательностью. Очевидно, что наибольшее время вычисления будет в том случае, когда все блоки имеют одинаковую длину, так как коль скоро восприняты два блока различной длины, то образование всех последующих выходов делается тривиальным. Напо-

мним далее, что 2^k есть функция, вычисляемая в реальное время [4]. Поэтому блоки в базе входной последовательности могут быть проверены на длину, записаны и пересчитаны в реальное время. Выход, связанный с базой последовательности, может быть образован с такой же быстротой, с какой поставляются входы. Поэтому в определении наибольшего времени вычисления нам нужно рассмотреть только лишь входные последовательности, содержащие более 2^k блоков. Заметим, наконец, что выходы, связанные с нулями и единицами хвостовых блоков, могут быть выработаны с такой же быстротой, с какой поступают соответствующие входные символы; единственно «трудные» выходы суть выходы в конце блоков.

Чтобы сравнить хвостовой блок с одним из записанных базовых блоков, машина должна просмотреть два набора, сравнивая их символ за символом. Если длина каждого блока есть k , то требуемое на это время состоит в точности из k единиц времени. Однако хвостовой блок может сравниваться со всеми записанными базовыми блоками, и читающая головка, которая рассматривает хвостовой набор, должна возвращаться к началу этого набора для следующего сравнения. Поэтому время, нужное для того, чтобы сравнить один хвостовой набор со всеми различными базовыми наборами, должно быть $2k2^k$. К этому времени мы должны добавить время, необходимое на возвращение соответствующей читающей головки к началу записанных базовых блоков, чтобы машина была в позиции для сравнения следующего хвостового блока. Таким образом, в общем мы имеем $2k2^k + (k+1)2^k < 3(k+1)2^k$ единиц времени для обработки каждого хвостового блока.

Если число хвостовых блоков есть x , то время, необходимое для записи базовой части входной последовательности и для сравнения каждого из хвостовых блоков, самое большее есть

$$T = (k+1)2^k + 3x(k+1)2^k.$$

С другой стороны, общее число символов во входной последовательности есть

$$\begin{aligned} n &= (k+1)2^k + x(k+1), \\ n^2 &= (k+1)^2 2^{2k} + 2x(k+1)^2 2^k + x^2(k+1)^2 > \\ &> (k+1)2^k + 3x(k+1)2^k = T. \end{aligned}$$

Поэтому число единиц времени, нужное для обработки входной последовательности длины n , всегда меньше чем n^2 , и множество A распознается с временной границей $T(n) = n^2$.

(Более детальные вычисления показывают, что A распознаваемо с границей $n^2/\log_2 n$, но граница n^2 достаточна для наших целей.)

В. Нижняя граница времени при распознавании множества A . Целью этого раздела является получение нижней границы времени вычисления, нужного для распознавания множества A . Для этого будет показано, что каждая машина, которая распознает A , должна тратить некоторое минимальное количество времени на переработку определенных входных последовательностей. Конечно, точное количество времени, нужное машине, зависит от ее параметров: числа лент, числа ленточных символов, которые она использует, и т. д. С другой стороны, мы знаем из предыдущей части, что множество A распознаваемо с временной границей $T(n) = n^2$, так что будет достаточно рассмотреть машины, которые работают с этой границей. Предположим поэтому, что нам дана машина M со входом, такая, что она

- а) распознает множество A с временной границей $T(n) = n^2$,
- б) имеет t лент, s ленточных символов и Q внутренних состояний.

Мы докажем, что вычисления на такой машине не могут быть выполнены с временной границей

$$\frac{Cn^2}{t \log s \log^2 n},$$

где C есть подходящая константа и n — длина входной последовательности.

При доказательстве этого результата удобно использовать обобщение понятия «состояние». Пусть **конфигурация** машины Тьюринга в данный момент означает:

- а) внутреннее состояние машины в данный момент;
- б) наборы символов, которые имеются на лентах машины в данный момент;
- с) позиции этих наборов по отношению к читающим головкам машины.

Таким образом, конфигурация машины в данный момент единственным образом определяет, как эта машина будет реагировать на последующую входную последовательность. Длина конфигурации есть максимальное расстояние между самым левым и самым правым непустыми символами по всем ленточным наборам. Заметим, что если машина работала t единиц времени, то длина ее конфигурации не может превышать t .

Чтобы найти нижнюю границу времени вычисления данной машины M , мы рассмотрим только лишь такие вычисления, которые машина выполняет на некоторых ограниченных множествах входных последовательностей A_1, A_2, A_3 и т. д. Каждый элемент множества A_k содержит в точности 2^{k+1} блоков длины k , разделенных единственной двойкой, т. е. каждый член A_k со-

держит базовый сегмент из 2^k блоков, за которыми следует хвостовой сегмент из 2^k блоков. Никакие ограничения не накладываются наборы из 0 и 1, которые присутствуют в базовых или хвостовых блоках. Например, последовательность 012002002102012102102012 относится к множеству A_2 .

Мы теперь хотим найти нижнюю границу максимума времени, которое машина M должна потратить при обработке индивидуальных элементов множества A_k , при помощи k, t, s, Q . Каждое вычисление, которое машина M выполняет на элементах A_k , можно разбить на две части: вычисление при обработке базы входа и вычисление при обработке хвоста. Мы будем игнорировать детали первой части и остановимся на конфигурации, в которой пребывает машина после первой части. Таким образом, мы можем думать о первой части вычисления как об акте отображения базового сегмента в элементе из A_k в такую конфигурацию машины, в которой машина M начинает обработку хвостовой части входной последовательности.

Конкретное отображение базовых сегментов на машинные конфигурации, в которых машина M может оказаться, сильно влияет на время, которое M при вычислениях должна потратить на хвостовые части. Таким образом, один из путей минимизации общего времени вычисления для членов из A_k есть «оптимизация» отображения, т. е. выбор отображения, для которого подходящим образом построенная машина при вычислении на хвостовых частях должна будет тратить времени как можно меньше. Конечно, само отображение должно быть таким, чтобы его можно было выполнить относительно быстро, именно с границей $T(n) = n^2$. Однако для наших целей удобнее рассмотреть отображения из более широкого класса, полностью игнорируя время, которое потребовалось бы машине Тьюринга для выполнения отображения. Выбор оптимального отображения из большего класса приведет к минимальному времени вычисления на хвостовых частях, которое по крайней мере так же мало, как и полученное при выборе отображения из меньшего класса «быстро реализуемых» отображений.

Рассмотрим теперь множество отображений базовых сегментов из A_k на машинные конфигурации, удовлетворяющих следующим условиям:

- 1) Каждая конфигурация указанного отображения совместна с машиной M , т. е. она включает t лент, s ленточных символов и Q внутренних состояний, используемых M .
- 2) Длина каждой конфигурации в отображении равна самое большее $4(k+1)^{2^{2k}}$ — максимальной длине конфигурации, которая может встретиться в процессе вычисления, выполняемого M на одном из элементов A_k .

3) Каждая конфигурация, которую дает отображение, есть конфигурация, в которой машина M работала бы «корректно» на некотором единственном хвостовом блоке. Это означает следующее. Предположим, что машина M находится в конфигурации, соответствующей некоторому произвольно выбранному базовому сегменту из A_k , и машине M подан произвольно выбранный хвостовой блок длины k . Тогда M должна дать на выходе k нулей, а затем 1 и 0 в соответствии с тем, есть ли данный хвостовой набор в данном базовом сегменте.

Время, которое нужно M , чтобы произвести $k + 1$ выходов, указанных в 3), зависит от конкретного базового сегмента и хвостового блока и конкретного отображения. Максимум этого времени по всем базовым сегментам из A_k и по всем хвостовым блокам длины k называется **временем сравнения**, связанным с данным отображением. Из определенного множества отображений выберем одно, имеющее наименьшее время сравнения; обозначим это отображение через F_k , а соответствующее время сравнения — через w_k . Заметим, что это отображение F_k определено существует. Только лишь конечное число отображений удовлетворяет 1) и 2), и только лишь конечное число входных последовательностей потребуется рассмотреть при проверке 3) и при определении минимального времени сравнения.

Следующим шагом является получение нижней границы w_k . Ее определяет следующая лемма¹⁾.

Лемма 1. $w_k > \frac{2^k - \log Q - t \log s - 1}{2t \log s}.$

Доказательство. Заметим сначала, что число машинных конфигураций, которые различаются или внутренними состояниями, или ленточными наборами, появляющимися в x квадратах по обеим сторонам от читающих головок машины, равно

$$Qs^{t(2x+1)},$$

где величина $t(2 + 1)$ — общее число занятых квадратов. В частности, если мы положим

$$x_k = \frac{2^k - \log Q - t \log s - 1}{2t \log s},$$

то $Qs^{t(2x_k+1)} = Qs^{(2^k - \log Q - 1)/\log s} = 2^{2^k - 1} < 2^{2^k} - 1.$

Распределим далее базовые сегменты из A_k по классам в соответствии с наборами из 0 и 1, которые входят в их блоки.

¹⁾ В этой работе все логарифмы берутся по основанию 2.

Так как существует 2^k различных наборов длины k , то число таких классов $2^{2^k} - 1$. Выберем по представителю из каждого класса и определим для каждого представителя машинную конфигурацию, в которую он отображается посредством F_k . По крайней мере два из этих представителей базовых сегментов должны отобразиться в конфигурации, у которых одинаковы внутренние состояния и наборы, появляющиеся в x_k квадратах под читающими головками, так как число конфигураций, которые отличаются только что указанными элементами, меньше чем $2^{2^k} - 1$. Обозначим эти два базовых сегмента через σ_1 и σ_2 .

Должен существовать по крайней мере один набор из 0 и 1, который находится в одном и только одном из сегментов σ_1 и σ_2 . Посмотрим, что случится, если M находится в конфигурации, соответствующей σ_1 или σ_2 , и на M подан один хвостовой блок, содержащий критический набор. Чтобы вычислить выход M для этого хвостового блока, нужно определить, соответствует ли конфигурация M сегментам σ_1 и σ_2 . Но такое определение необходимо влечет за собой проверку по крайней мере одного символа, который будет удален более чем на x_k квадратов от читающей головки. Так как время, нужное для M , чтобы передвинуть такой символ под соответствующую читающую головку, превышает x_k , то мы установили $w_k > x_k$. Ч. т. д.

Напомним, что отображения базовых сегментов в машинные конфигурации, которые машина M проходит, включаются в отображения, из которых было выбрано F_k . Но по определению F_k есть отображение, для которого максимум количества времени, которое M должна потратить на первом хвостовом блоке, наименьший. Таким образом, если машина M использует свое собственное отображение, чтобы подготовиться для проверки хвостовых блоков, то она в самом худшем случае не может иметь на первом хвостовом блоке меньше времени вычисления, чем в случае с F_k . Другими словами, должна существовать входная последовательность в множестве A_k , на которой машина M потратит по крайней мере w_k единиц времени на первом хвостовом блоке.

Интуитивно кажется, что процесс сравнения одного блока в хвосте последовательности в сущности не зависит от процесса сравнения следующего блока и что наихудший случай вычисления времени безотносителен к позиции хвостового блока. Чтобы проверить это предположение, мы должны доказать, что в процессе сравнения блоков в начале хвоста M не может переупорядочить свои записи таким образом, чтобы ускорить сравнение более дальних блоков. Основой доказательства будет то, что если бы такое переупорядочение было возможно, то можно

было бы найти отображение, которое «лучше», чем F_k . (Читатель, который предпочитает поверить интуиции, в этом месте может просто прочесть утверждение леммы 2, а потом перейти к теореме 1.)

Доказательство облегчается, если члены из A_k изображаются деревом, по одному пути в дереве для каждого члена из A_k . Так

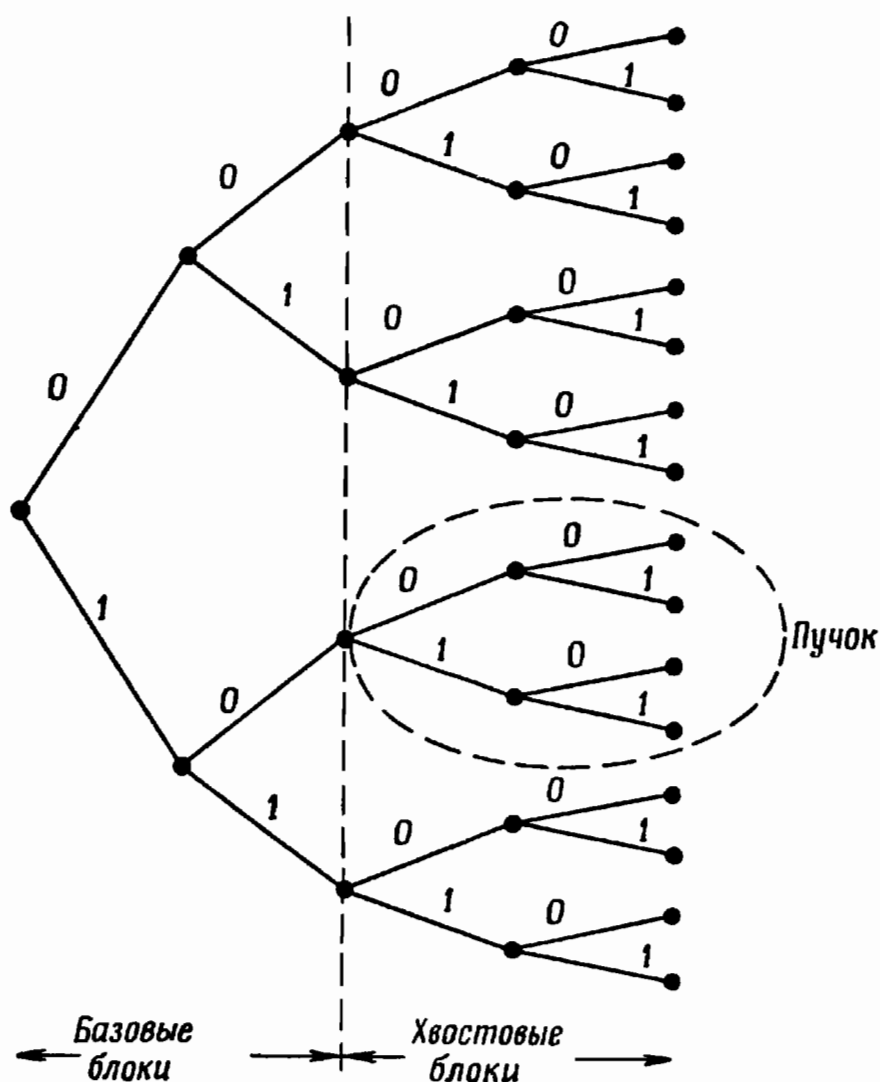


Рис. 1. Дерево для множества A_1 .

как каждая последовательность из A_k состоит из 2^{k+1} блоков, каждый из которых может содержать любой из 2^k наборов, то соответствующее дерево содержит $2^{k+1} + 1$ уровней точек с 2^k ветвями, исходящими из каждой неконечной точки. Рис. 1 показывает дерево для случая, в котором $k = 1$, когда существуют в точности два возможных набора блоков (0 и 1), следовательно, две ветви исходят из каждой неконечной точки. Удобно разделить те ветви в дереве, которые соответствуют базовым блокам, от тех, которые соответствуют хвостовым блокам, что сделано пунктирной вертикальной линией на рис. 1. Поддерева, которые исходят из точек на отделяющей линии, называются пучками дерева;

каждый пучок соответствует в точности одному из 2^k возможных базовых сегментов.

Лемма 2. *Существует по крайней мере одна входная последовательность в множестве A_k , на которой машина M тратит по крайней мере $2^k w_k$ единиц времени на хвостовой части последовательности.*

Доказательство. Рассмотрим каждый член из A_k , подадим его на машину M и заметим, сколько времени тратит M на каждом блоке в хвосте последовательности. Если M тратит меньше чем w_k единиц времени на данном блоке, то отметим соответствующую ветвь в дереве знаком минус, в противном случае соответствующую ветвь пометим знаком плюс. Если все ветви, исходящие из конкретной точки, помечены знаком минус, то назовем эту точку **минус-точкой**, в противном случае — **плюс-точкой**.

Сначала мы должны показать, что не все пучки содержат минус-точку. Заметим, что каждая точка в пучке соответствует конфигурации M , в частности конфигурации, в которой M должна получить символы хвостового блока и вычислить соответствующий этому блоку выход. Если все ветви, исходящие из некоторой точки пучка, помечены знаком минус, то конфигурация, соответствующая этой точке, должна быть такой, чтобы машина потратила меньше чем w_k единиц времени на следующем хвостовом блоке вне зависимости от набора, содержащегося в этом блоке.

Таким образом, если каждый пучок содержит минус-точку, то можно определить новое отображение F'_k , которое имеет время сравнения, меньшее, чем F_k . Рассмотрим каждый базовый сегмент, заметим пучок в дереве, которому он соответствует, и найдем последовательность хвостовых блоков, которая ведет к минус-точке в этом пучке. Затем подадим базовый сегмент M , за которым следует соответствующая последовательность хвостовых блоков, и заметим конфигурацию, в которой M оказалась. Пусть эта конфигурация будет конфигурацией, поставленной в соответствие базовому сегменту при новом отображении F'_k .

Отображение F'_k , безусловно, удовлетворяет условиям 1) и 3). Оно удовлетворяет также условию 2), так как конфигурации, получающиеся в результате этого отображения, суть конфигурации, которые M достигает в процессе вычисления на входной последовательности длины $2(k+1)2^k$ или меньшей. Так как M работает с временной границей n^2 , то длина такой конфигурации не может превышать $4(k+1)^2 2^{2k}$. Таким образом, F'_k относится к множеству отображений, из которого было вы-

брано F_k . Время сравнения для F'_k связано с минус-точками и, следовательно, по определению меньше, чем ω_k — время сравнения для F_k . Но в этом есть противоречие, так как F_k было выбрано с наименьшим временем сравнения. Поэтому предположение о том, что каждый пучок может иметь минус-точку, невыполнимо, и некоторый пучок поэтому должен иметь только лишь плюс-точки.

Но если данный пучок имеет только лишь плюс-точки, то каждая точка в пучке должна иметь по крайней мере одну исходящую ветвь, которая отмечена знаком плюс. Поэтому в пучке должен существовать по крайней мере один путь, который включает в себя ветви, отмеченные только плюсом. Выберем в A_k член, базовый сегмент которого соответствует пути от корня дерева до указанного пучка и хвостовой сегмент которого соответствует пути в пучке, который содержит только лишь отмеченные плюсом ветви. Этот член из A_k таков, что M тратит по крайней мере ω_k единиц времени на каждом его хвостовом блоке и, следовательно, в общем по крайней мере $2^k \omega_k$ единиц времени на всем хвостовом сегменте. Ч. т. д.

Результаты лемм 1 и 2 можно теперь объединить и дать нижнюю границу времени вычисления, необходимого для распознавания множества A .

Теорема 1. Пусть M есть машина со входом, имеющая t лент, s ленточных символов и распознающая множество A . Пусть $T(n)$ означает максимум количества времени, которое M тратит на всякой входной последовательности длины n . Тогда $T(n)$ превышает

$$\frac{n^2}{48t \log s \log^2 n}$$

для достаточно больших значений n .

Доказательство. Нам нужно рассмотреть только лишь случай, в котором M работает с границей n^2 , так как иначе утверждение теоремы будет безусловно правильным. Пусть n_k означает общую длину последовательностей в множестве A_k , и пусть T_k означает максимум времени, которое M должна потратить на последовательностях из A_k . Тогда для всякого данного значения n выберем k таким, чтобы $n_k \leq n \leq n_{k+1}$. Так как время вычисления, нужное машине M , монотонно по n , то $T(n) \geq T_k$, в то время как по леммам 2 и 1 мы имеем, что

$$T_k \geq 2^k \omega_k > \frac{2^k (2^k - \log Q - t \log s - 1)}{2t \log s}.$$

Таким образом, если n , а следовательно, и k достаточно велики, то

$$T_k > \frac{2^{2k}}{3t \log s}.$$

Заметим теперь, что $n_{k+1} = 2(k+2)2^{k+1}$, и поэтому $\log n_{k+1} > k+2$. Следовательно,

$$T(n) \geq T_k > \left[\frac{4(k+2)^2 2^2}{4(k+2)^2 2^2} \right] \frac{2^{2k}}{3t \log s} > \frac{n_{k+1}^2}{48t \log s \log^2 n_{k+1}}.$$

Так как $n_{k+1} > n > e$, то дробь $n_{k+1}/\log n_{k+1}$ превысит дробь $n/\log n$. Поэтому

$$T(n) > \frac{n^2}{48t \log s \log^2 n}. \quad \text{Ч. т. д.}$$

С. Обсуждение. Теорема 1 дает нижнюю границу порядка роста времени распознавания множества A , причем время есть функция от n , t , s . Чтобы увидеть, является ли порядок роста разумным, вспомним, что число b существенно различных базовых частей членов множества A_k есть $2^{2^k} - 1$, в то время как общая длина n каждого члена из A_k есть $(k+1)2^{k+1}$. Число битов, необходимых для выяснения, который из b существенно различных базовых сегментов присутствует, приблизительно равно 2^k , а это приблизительно равно $n/\log n$. Но если рассматриваемая машина использует t лент и s ленточных символов, то максимальное число квадратов на ленте, нужное для записи этого сегмента, приблизительно равно $n/(t \log s \log n)$. Это число квадратов есть максимальное число сдвигов ленты, которые должны быть сделаны, чтобы сравнить один хвостовой блок.

Общее число хвостовых блоков в члене из A_k есть 2^k , или приблизительно $n/\log n$. Теорема 1 устанавливает, что общее время вычисления в худшем случае должно быть пропорционально произведению числа хвостовых блоков, подлежащих сравнению, и максимального числа квадратов на ленте, нужных для записи всей существенной информации о базовом сегменте. Таким образом, невозможно улучшить вполне очевидный способ записи всех наборов, которые присутствуют в базовом сегменте, а потом просмотреть эту запись для каждого вновь поступившего хвостового блока.

Порядок роста, указанный в теореме 1, меньше порядка, полученного в разд. 2, А, в $\log^2 n$ раз. Это расхождение возникает потому, что машина, описанная в разд. 2, А, использует несколько неэффективный метод записи базовых наборов. Вместо просто записи наборов одного за другим более экономно записать их в «факторной» форме, удаляя общий начальный сегмент у двух или более наборов, где это возможно. Тогда пара набо-

ров 0000 и 0001 может быть записана как $000(0 + 1)$, множество 0000, 0001, 0101, 1000, 1010 может быть записано как $0(00(0 + 1) + 101) + 10(00 + 10)$. Используя такую схему, можно построить машину, которая распознает множество A и работает с временем, пропорциональным $n^2/(t \log s \log^2 n)$ для достаточно больших t и s ¹⁾. Таким образом, оптимальное время вычисления из теоремы 1 есть с точностью до постоянного множителя лучшее достижимое время.

«Время поиска» при сравнении одного хвостового блока так сильно влияет на время распознавания множества A , что желательны другие вычисляющие модели, в которых время поиска можно снизить. Одна возможность, которая будет исследована в разд. 3, предполагает машину Тьюринга с двумерными лентами. Так как запас данного числа битов информации может быть помещен на двумерной ленте «диаметра», много меньшего «диаметра» одномерной ленты, то и время исследования при использовании двумерной ленты может быть сделано много меньше. Уменьшение времени поиска снизило бы порядок роста общего времени вычисления.

3. ДВУМЕРНЫЕ ЛЕНТЫ

В этой части мы рассмотрим машины Тьюринга со входом, снабженные одной (или более) бесконечной двумерной лентой. Назовем такую машину для простоты двумерной машиной. На каждом шаге вычисления двумерная машина может написать новые символы в обозреваемых квадратах, сдвинуть каждую из своих лент влево, вправо, вверх или вниз (или остаться на месте) и изменить внутреннее состояние. Как и в обычных машинах со входом, двумерная машина со входом начинает вычисления с пустыми лентами. Затем подаются входные последовательности, вырабатываются выходные последовательности, в точности как и в случае одномерных лент. В частности, определение времени вычисления остается неизменным.

В предыдущем разделе предполагалось, что использование двумерных лент может уменьшить количество времени, нужного для распознавания множества A . В самом деле, можно построить двумерную машину, которая распознает A с временем, пропорциональным $n^{3/2}$, по сравнению с порядком роста n^2 у одномерной машины. Таким образом, время, необходимое для двумерной реализации, пропорционально степени $3/4$ лучшего времени, получающегося при одномерной реализации. Вместо

¹⁾ Грубая оценка показывает, что константа пропорциональности не более 20 или 30, хотя, без сомнения, ее можно сделать много лучше.

двумерной реализации A мы займемся проблемой распознавания, для которой разница между одно- и двумерным временами вычислений даже больше, чем для множества A .

Множество исследуемых в этом разделе входных последовательностей лучше всего описывается в терминах поведения распознающей его двумерной машины. Эта машина M_2 работает на входных символах U, D, L, R, W и E , которые означают команды *вверх, вниз, влево, вправо, написать* и *проверить* соответственно. Машина имеет одну двумерную ленту и использует ленточные символы 0, 1. В начале вычисления каждый квадрат ленты содержит 0.

При получении одного из входных символов U, D, L или R машина M_2 сдвигает читающую головку на один квадрат в указанном направлении, т. е. она сдвигает ленту на один квадрат в противоположном направлении, и вырабатывает выходной символ 0. При получении входного символа W машина не перемещает ленту, но пишет символ 1 в обозреваемом квадрате и вырабатывает выход 0. При получении входного символа E машина ничего не делает с лентой, но вырабатывает выход 0 или 1 в соответствии с тем, содержит ли обозреваемый квадрат 0 или 1. Например, если M_2 начинает с ленты, заполненной нулями, а на ее вход подана последовательность $RWURWDWLEDERUER$, то она выработает выходную последовательность 000000001000010.

Множество входных последовательностей, распознаваемых M_2 , обозначим через B . Согласно определению, B распознаваемо двумерной машиной с границей времени $T(n) = n$. Согласно Хартманису и Стирнзу [2, 3], переход от двумерной реализации к одномерной требует самое большее квадрата времени вычисления. Таким образом, можно построить одномерную машину, которая распознает множество B с временной границей $T(n) = n^2$. Теперь мы желаем найти нижнюю границу времени вычисления одномерной машины, которая распознает B .

Для этой цели удобно рассмотреть некоторые множества B_1, B_2, B_3 и т. д. входных последовательностей, определенные в терминах поведения двумерной машины M_2 на этих множествах. Каждый член множества B_n имеет две части: *пишущий сегмент* и следующий за ним *читающий сегмент*. Пишущий сегмент есть последовательность, состоящая из U, D, L, R, W . На нем M_2 двигает читающую головку вдоль «спирального» пути, как это показано на рис. 2, записывая 1 в некотором произвольно выбранном подмножестве квадратов, которые она посещает. Когда пройдено k^2 квадратов (необязательно с записями), читающая головка возвращается наикратчайшим путем к исходной точке, или к *дому-квадрату*.

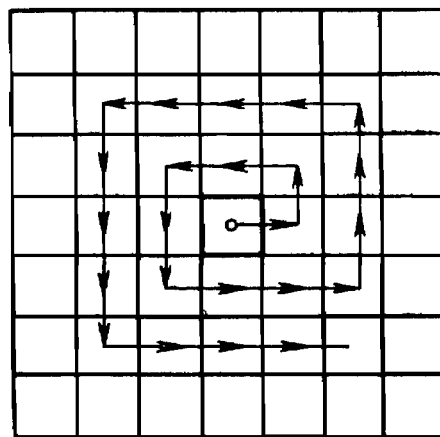
Читающий сегмент каждого члена из B_k делится на k последовательных кусков, называемых запросами. Каждый запрос состоит из трех частей:

1) последовательности из U, D, L, R , на которой M_2 двигает читающую головку наикратчайшим путем из дома-квадрата к одному из квадратов, посещенному на читающем сегменте;

2) одного E , на котором M_2 вырабатывает на выходе символ, записанный в указанном квадрате;

3) последовательности из U, D, L, R , на которой M_2 двигает читающую головку наикратчайшим путем назад в дом-квадрат.

Таким образом, на запрашивающей последовательности $UUURRRREDDDLLLL$ машина M_2 проверяет содержимое квад-



Р и с. 2. Путь на пишущем сегменте.

рата, который расположен на три квадрата выше и на четыре квадрата правее дома-квадрата. Точный путь следования к запрашиваемому квадрату и от него неважен. Любой из k^2 допустимых квадратов может быть проверен данным запросом.

Число входных символов, необходимых для перемещения читающей головки машины M_2 вдоль спирального пути, равно $k^2 - 1$; число символов, необходимых для написания в некотором подмножестве посещенных квадратов единиц, равно самое большее k^2 ; число символов, нужных для возвращения читающей головки в дом-квадрат, равно $k - 1$. Таким образом, общее число символов в читающем сегменте элемента B_k менее $3k^2$. Число входных символов, необходимых для одного запроса, равно самое большее $2k - 1$, так как любой квадрат, который посещался на читающем сегменте, достигается из дома-квадрата в $k - 1$ или меньше шагов. Таким образом, число символов в читающем сегменте члена из B_k менее $2k^2$, и общее число символов в члене из B_k менее $5k^2$.

Предположим теперь, что M_1 есть одномерная машина, распознающая множество B . Как велико время вычисления, используемое машиной M_1 , если она имеет t лент, s ленточных

символов и Q внутренних состояний? Так как известно, что можно построить одномерную машину, распознающую B с временной границей n^2 , то можно допустить, что M_1 работает именно с этой границей. Далее можно использовать рассуждения, близкие к рассуждениям разд. 2, В, чтобы показать, что существует некоторое множество B_k , на котором M_1 должна потратить порядка k^3 единиц времени. Из этого следует, что M_1 требует времени вычисления, пропорционального $n^{3/2}$.

Рассмотрим сначала отображения пишущих сегментов в членах из B_k на одномерно-машинные конфигурации. В частности, рассмотрим такие отображения, которые удовлетворяют каждому из следующих ограничений:

4) Каждая машинная конфигурация, порожденная отображением, совместна с M_1 , т. е. она включает t лент, s ленточных символов, используемых M_1 , и Q внутренних состояний M_1 .

5) Длина каждой конфигурации равна самое большее $25k^4$ — максимальной длине конфигурации, возможной в процессе вычисления, которое выполняет M_1 на элементах B_k .

6) Каждая конфигурация есть конфигурация, в которой M_1 работала бы «корректно» на любой единичной запрашивающей последовательности. Это означает следующее. Предположим, что M_1 помещена в конфигурацию, соответствующую конкретному пишущему сегменту из B_k , и на вход подается произвольно выбранная запрашивающая последовательность. В ответ на E из этой последовательности M_1 должна выработать выход 1 или 0 в соответствии с тем, записана ли 1 в проверяемом квадрате при данном пишущем сегменте или не записана.

Выберем из этих отображений одно, для которого максимальное время, которое M_1 должна потратить на произвольном запросе, описанном в 6), как можно меньше. Обозначим это «лучшее» отображение через G_k , а соответствующее максимальное «время запроса» через v_k .

Рассуждениями, аналогичными рассуждениям леммы 1, можно показать, что

$$v_k > y_k = \frac{k^2 - t \log s - \log Q - 1}{2t \log s}.$$

Чтобы это показать, заметим, что число конфигураций, которые различаются или внутренними состояниями, или символами, появляющимися в y_k квадратах с той или другой стороны от различных читающих головок M_1 , есть

$$Qs^{t(2y_k+1)} = 2^{k-1}.$$

С другой стороны, существует 2^k различных наборов из 0 и 1, которые могут быть составлены на ленте M_2 при читающем сег-

менте из множества B_k . Таким образом, G_k должно отображать некоторые два пишущих сегмента, которые порождают различные наборы на ленте M_2 , в конфигурации M_1 , которые согласуются по внутренним состояниям и по ленточным символам, появляющимся в y_k квадратах под читающими головками. Поэтому должен быть возможен выбор конкретного пишущего сегмента и первого запрашивающего сегмента, для которых M_1 должна проверить некоторый ленточный символ, лежащий дальше чем на y_k квадратов от читающей головки. Так как M_1 должна потратить по крайней мере $y_k + 1$ единиц времени, чтобы достичь такого символа, то максимальное время запроса при отображении G_k превышает y_k .

Следующим шагом — рассуждениями, аналогичными рассуждениям леммы 2, — должно быть показано, что найдется некоторый элемент из B_k , на котором M_1 должна потратить по крайней мере kv_k единиц времени, вырабатывая выходные символы, связанные с читающим сегментом последовательности. Опять удобно представить элементы B_k в виде дерева. Это дерево имеет 2^{k^2} «главных ветвей», по одной для каждого из различных пишущих сегментов в элементах B_k . На конце каждой основной ветви есть поддерево, или пучок, содержащий k уровней ветвей. Из каждой точки в каждом пучке исходят k^2 ветвей, по одной для каждого из различных запросов, которые могут быть сделаны в каждой части читающего сегмента в последовательности из B_k .

Следуя рассуждениям леммы 2, подадим каждую последовательность из B_k на машину M_1 и определим длину промежутка времени, которое M_1 тратит на переработку каждой запрашивающей последовательности в читающем сегменте. Затем пометим каждую ветвь в каждом пучке дерева знаками плюс или минус в соответствии с тем, тратит или не тратит M_1 v_k единиц времени на соответствующем запросе. Если бы в каждом пучке имелась по крайней мере одна точка, из которой исходили бы только лишь минус-ветви, то можно было бы найти отображение G'_k , которое удовлетворяет условиям 4), 5) и 6) и максимальное время запроса которого было бы менее v_k . Но так как выбор G_k исключает существование такого G'_k , то должен существовать хотя бы один пучок, содержащий только лишь плюс-точки. Поэтому должен существовать хотя бы один член из B_k , на котором M_1 тратит по крайней мере kv_k единиц времени на читающем сегменте.

Наконец, рассуждения, аналогичные рассуждениям теоремы 1, приводят к нижней границе $T_1(n)$ при вычислении времени, нужного M_1 . Напомним, что число символов в каждом

члене из B_k меньше чем $5k^2$. Тогда если дано произвольное значение n , то выберем k так, чтобы

$$5k^2 \leq n < 5(k+1)^2$$

и, следовательно, чтобы

$$n^{3/2} < 100k^3.$$

Так как $T_1(n)$ монотонно возрастает, то $T_1(n)$ по крайней мере так же велико, как и максимальное время, которое M_1 должна потратить на читающих сегментах в членах из B_k . Таким образом,

$$T_1(n) \geq kv_k > \frac{k(k^2 - t \log s - \log Q - 1)}{2t \log s}.$$

Если n , а следовательно, и k достаточно велики, то

$$T_1(n) > \frac{k^3}{3t \log s} > \frac{n^{3/2}}{300t \log s}.$$

Таким образом, $T_1(n)$ растет пропорционально $n^{3/2}$. С другой стороны, время $T_2(n)$, необходимое для распознавания множества B двумерной машиной, растет не быстрее n . Выражая $T_1(n)$ через $T_2(n)$, имеем

$$T_1(n) > \alpha (T_2(n))^{3/2},$$

где α есть константа при конкретных значениях t и s . Другими словами, по крайней мере для проблемы распознавания множества B уменьшение размерности ленты с двойки до единицы требует увеличения порядка роста времени вычислений по крайней мере степени $3/2$.

4. РАЗМЕРНОСТЬ И СКОРОСТЬ

А. Обобщения множества B . Методы предыдущих разделов могут быть применены к машинам с трехмерными и более лентами. В частности, определение множества B можно расширить на большее число измерений. Рассмотрим m -мерную машину M_m , аналогичную машине M_2 ; M_m способна сдвигать ленты во всех $2m$ направлениях, записывая 1 в «квадратах», которые она посещает, и проверяя символы, которые она обзорекает. Пусть такой машине подаются входные последовательности в алфавите $\{d_{1+}, d_{1-}, \dots, d_{m+}, d_{m-}, W, E\}$. На символах d_{i+} и d_{i-} машина сдвигает ленту параллельно i -й оси, а на W и E пишет 1 или соответственно проверяет квадрат, который она в данный момент обзорекает, на присутствие 1. Как и раньше, машина вырабатывает выход 1 только лишь в ответ на вход E и

только лишь в случае, если квадрат, обозреваемый в настоящий момент, содержит 1.

Пусть множество входных последовательностей, распознаваемых M_m , обозначено через W^m . (Так, множество B идентично множеству W^2 .) Множество W^m распознаваемо m -мерной машиной в реальное время. Но какое время нужно машине с меньшим числом измерений, чтобы распознать его? Пусть $T_j^m(n)$ означает наименьшее время, в которое W^m распознаваемо j -мерной машиной. Тогда методами предыдущего раздела может быть показано, что

$$7) \quad T_j^m(n) \geq C (T_m^m(n))^{2-j/m},$$

где C есть функция от числа лент и ленточных символов, используемых j -мерной машиной.

Рассуждения, приводящие к этому результату, могут быть обобщены следующим образом. Определим сначала множества $W_1^m, W_2^m, \dots, W_k^m, \dots$, аналогичные множествам B_k разд. 3. Каждый член из W_k^m начинается с пишущего сегмента, на котором M_m посещает каждый квадрат m -мерного гиперкуба со стороной k , и пишет 1 в некоторых из этих квадратов. За этим пишущим сегментом следует читающий сегмент, в котором запрашиваются в точности k^{m-1} квадратов гиперкуба.

Применим далее к последовательности из W_k^m некоторую j -мерную машину M , которая распознает W^m . Из результатов разд. 3 вытекает, что наихудшее время вычисления для одного запроса по крайней мере пропорционально k^{m-j+1} и что наихудшее время вычисления всех k^{m-1} запросов по крайней мере пропорционально k^{2m-j} . С другой стороны, время, необходимое для обработки членов из W_k^m на M_m , самое большее пропорционально k^m . Таким образом, время, необходимое j -мерной машине, по крайней мере пропорционально степени $(2m-j)/m$ времени, необходимого m -мерной машине.

Модификации машин M_m приводят к определению множеств, для которых возможна данная в 7) нижняя временная граница. Таким образом, существуют проблемы распознавания, для которых необходимое и достаточное время при различных измерениях может быть определено с точностью до постоянных множителей. Более того, порядок роста времени вычисления модифицированных форм множества W^m увеличивается с каждым уменьшением размерности ленты от m до 1.

В. Заключение. Эта работа служит двум целям. Первая — дать метод нахождения нижних границ при вычислении вре-

мени в некоторых проблемах распознавания со входом. Проблемы, для которых этот метод оказывается полезным, суть те, в которых информация задается машине сначала в первой части входной последовательности, а потом эта информация запрашивается у машины другой частью входной последовательности. Таким образом, машина должна запасти или записать поступающую информацию таким образом, чтобы потом ее можно было быстро найти по требованию.

Рассмотренный здесь метод не принимает во внимание проблему записи входной информации; внимание концентрируется на проблеме нахождения ранее запасенной информации. Здесь допускается, что информация записана самым лучшим образом, т. е. так, чтобы минимизировать время, нужное для нахождения произвольного куска информации позже. Время поиска, полученное с использованием этого допущения, очевидно, не будет превосходить времени поиска, которое может быть получено, если используется другая форма записи. Таким образом, мы можем потратить столько времени, сколько необходимо на предварительную обработку запасенной информации, если это позволит нам ограничить время поиска.

Заметим, что предположение об оптимальной предварительной обработке в действительности дает нам две вещи. Во-первых, это приводит к нижней границе времени, нужного для нахождения одной порции запасенной информации (одного набора в случае множества A или одного квадрата ленты в случае множества B). Эта граница основана на числе различных «предысторий», которые требуется запомнить, и на числе квадратов ленты, нужных, чтобы их изобразить. Во-вторых, это приводит к заключению, что поиск каждой из последовательных порций информации в действительности является независимой проблемой и что наихудшее время вычисления одинаково для каждого из последовательных запросов.

Таким образом, в дополнение к нахождению нижних границ времени, нужного для распознавания множеств A и B , метод разд. 2 и 3 может оказаться полезным для других проблем с теми же самыми характеристиками. Эти характеристики в общем таковы: 1) вычисления со входом; 2) необходимость нахождения произвольных порций информации из произвольно длинного списка порций, поданного ранее; 3) независимость между последовательно запрашиваемыми порциями.

Вторая цель этой работы — исследовать связь между размерностью ленты и временем вычисления. Работы Хартманиса и Стирнза [2, 3] показывают, что уменьшение размерности ленты машины Тьюринга никогда не требует больше чем квадрата обычного времени вычисления. Неясно, однако, требует ли

в действительности уменьшение размерности ленты квадрата времени вычисления. Этот вопрос очень уместен, так как он показывает, что уменьшение числа лент у машины Тьюринга не влечет существенного увеличения времени вычисления в предположении, что ленты одномерны и число лент не меньше двух [5].

Еще не ясно, необходим ли квадрат. Результаты разд. 3 и 4, А показывают, что существенное увеличение времени вычисления необходимо по крайней мере для некоторых проблем, когда число измерений уменьшается. В частности, если число измерений уменьшается от m до j , то время вычисления может вырасти в степень $(2m - j)/m$. Таким образом, если число измерений сильно уменьшается, то время вычисления должно увеличиться в степени, близкой к 2.

Итак, имеем, что добавление каждого нового измерения дает иногда уменьшение времени вычисления, по крайней мере для некоторых проблем. Для обсуждаемых здесь примеров этот выигрыш уменьшается с каждым новым измерением, но не ясно, всегда ли это будет иметь место. Понятно то, что размерность ленты и количество лент влияют на время вычисления по-разному. Если увеличение числа измерений может существенно уменьшить порядок роста времени вычисления, то увеличение числа (одномерных) лент свыше двух не меняет существенно порядка роста времени вычисления.

ЛИТЕРАТУРА

1. Hennie F. C., One-tape, off-line Turing machine computations, *Information and Control*, 8 (1965), 553—578. (См. стр. 223—248 настоящего сборника.)
2. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, 117 (1965), 285—306 (Русский перевод: Кибернетический сборник (новая серия), вып. 4, «Мир», 1967.)
3. Hartmanis J., Stearns R. E., Computational complexity of recursive sequences, 1964 Proc. of the Fifth annual Symp. on Switching Circuit Theory and Logical Design, p. 82.
4. Yamada H., Real-time computation and recursive functions not real-time computable, *IRE Trans. on Electronic Computers*, EC-11 (1962), 753—760 (См. стр. 139—155 настоящего сборника.)
5. Hennie F., Stearns R. E., Two-tape simulation of multitape Turing machines, *JACM*, 13, № 4 (1966), 535—546. (См. настоящий сборник стр. 194—212.)

РАСПОЗНАВАНИЕ НА МАШИНЕ ТЬЮРИНГА СО ВХОДОМ ¹⁾

П. Стрнад

ВВЕДЕНИЕ

Многоленточная машина Тьюринга со входом M начинает вычисление в фиксированном состоянии s_0 (принадлежащем к алфавиту состояний) с пустыми лентами. Входные символы M образуют входной алфавит, выходные символы — выходной алфавит. В ответ на входной символ машина вырабатывает соответствующий выходной символ. Такой ответ не обязательно должен следовать непосредственно вслед за восприятием соответствующего входного символа: машина может предварительно совершить несколько шагов в процессе своей работы.

В это время (в течение подготовки ответа) машина не воспринимает каких-либо других входных символов. Входной символ с номером $i+1$ воспринимается машиной в момент времени, следующий за тем, в который выдается i -й выходной символ. Выходным алфавитом машины является $\{0, 1\}$, т. е. можно рассматривать машину как устройство, распознающее множество таких входных последовательностей, для которых последний символ в соответствующей выходной последовательности есть 1.

На каждом шаге вычисления машина может изменять символы, находящиеся под ее головками, сдвигать каждую из своих головок на один квадрат вправо или влево и изменять свое состояние. Общее число шагов, которые машина должна совершить для того, чтобы получить выходную последовательность, соответствующую данной входной последовательности, называется временем вычисления для этой входной последовательности.

Если время вычисления для каждой входной последовательности длины n меньше или равно $T(n)$ (где $T(n)$ — заданная функция), то говорят, что машина работает за время $T(n)$. Множество слов B тогда и только тогда распознается за время $T(n)$, когда существует машина Тьюринга со входом M , такая, что

- (а) M распознает множество слов B ,
- (б) M работает за время $T(n)$.

¹⁾ Strnad P., On-line Turing machine recognition, *Information and Control*, 12, № 5/6 (1968), 442—452.

Пусть машина со входом M распознает множество слов S ; тогда введем следующие обозначения:

$$\tau_C^M(n) = \max_{|u|=n} t(u),$$

где $t(u)$ — время вычисления для входной последовательности u , а $|u|$ — длина слова u во входном алфавите машины M .

Функции, вычислимые в реальное время¹⁾. Временная функция $V(n)$ называется вычислимой в реальное время в том и только в том случае, когда существует константа K и многоленточная машина Тьюринга M с записью на ленте, такая, что если машина M начинает свою работу при заполненной n символами одной из своих лент, то

(а) в течение самое большее $K \cdot V(n)$ шагов машина заканчивает свою работу;

(б) после остановки машины на одной из ее лент записано $V(n)$ символов.

Множество A . Слова множества A состоят из букв алфавита $\{0, 1, *\}$. Слово длины $i+1$, заканчивающееся $*$ и состоящее из 0 и 1, будем называть i -блоком ($i=1, 2, \dots$).

Пример. 001* является 3-блоком, 11010* это 5-блок. Последовательность из 2^i i -блоков будем называть i -базой.

Определим множество $A^{(i)}$: $w \in A^{(i)}$ тогда и только тогда, когда существует $k \geq 1$, такое, что w является последовательностью из $2^i + k$ i -блоков и последний в слове w i -блок совпадает с одним из первых 2^i i -блоков слова w . Тогда

$$\bigcup_{i=1}^{\infty} A^{(i)}$$

есть множество A из статьи Хенни [1].

Хенни [1] показал, что для любой машины Тьюринга M со входом, распознающей множество A , имеет место неравенство $\tau_A^M(n) > n^2 / (48 t \log s \log^2 n)$ для достаточно больших значений n (t — число лент, а s — число рабочих символов машины M).

Заметим, что повсюду в этой статье под $\log$ подразумевается $\log_2$.

¹⁾ Приведенное ниже определение отличается от определения функции, вычислимой в реальное время, данного в статьях Ямада, Рабина и др из настоящего сборника. Определенный здесь класс функций более широк — Прим. ред

1. МНОЖЕСТВО A_U И ЕГО РАСПОЗНАВАНИЕ НА МАШИНЕ ТЮРИНГА СО ВХОДОМ

Введем следующие обозначения:

$$R(n) \underset{1}{\leq} S(n), \quad \text{если} \quad \limsup_{n \rightarrow \infty} R(n)/S(n) < \infty;$$

$$R(n) \underset{2}{\leq} S(n), \quad \text{если} \quad \limsup_{n \rightarrow \infty} R(n)/S(n) < 1.$$

Определение. Пусть $U(n)$ — монотонно возрастающая функция, вычислимая в реальное время, причем $n \underset{2}{\leq} U(n) \underset{1}{\leq} n^{2-\varepsilon}$ для подходящего $\varepsilon > 0$. Тогда множество слов A_U определяется следующим образом:

(1) $A_U \subset A$, где A является множеством, рассмотренным во введении.

(2) $w \in A_U \cap A^{(i)}$ тогда и только тогда, когда длина w не больше чем $2^{i+1} \cdot (i+1)$ и на первых $[f(i)]$ позициях каждого i -блока w стоят произвольные символы, а на остальных $i - [f(i)]$ позициях i -блока записаны только 0, причем $f(i)$ определяется равенством $2^{f(i)} = U(2^i(i+1))/2^i$, откуда $f(i) = \log U(2^i(i+1)) - i$.

Замечание. Если r — действительное число, тогда $[r]$ обозначает целое число, такое, что $[r] \leq r < [r] + 1$.

Теорема 1. Для каждого множества слов A_U можно построить машину Тьюринга со входом M_U , распознающую множество A_U за время $C \cdot U(n)$, где C — соответствующая константа.

Доказательство. Опишем машину M_U , распознающую множество A_U . Пусть i -слово ($i = 1, 2, \dots$) является словом, состоящим из k i -блоков, где k — конечное число ($k = 0, 1, 2, \dots$). Если задано натуральное число i_0 , то все j -слова из A_U , где $j = 1, 2, \dots, i_0$, могут быть распознаны конечным автоматом. Таким образом, можно рассматривать только i -слова, где $i > i_0$. Одной из частей машины M_U будет устройство (многоленточная машина Тьюринга с записью на ленте), которое вычисляет в реальное время функцию $U(n)$, причем если это устройство начинает свою работу, используя записанные на одной из лент машины M_U n символов, то эта работа должна закончиться за время, не превосходящее $C_1 U(n)$, после чего на другой ленте M_U должны быть записаны $U(n)$ символов.

Работу, выполняемую машиной M_U , можно разбить на несколько этапов.

Первый этап начинается с начала ввода. В течение этого этапа:

(а) Машина определяет длину первого блока и отмечает эту длину на ленте T_1 . В процессе работы машины происходит

сравнение длины блоков входного слова с длиной первого блока. Если какой-то блок имеет другую длину, машина будет печатать на выходной ленте 0 в ответ на каждый последующий символ.

(б) Как только машина определяет длину $i+1$ первого блока, она начинает записывать на двух лентах T_2 и T_3 группы из $1, 2, 4, \dots, 2^{i-1}$ символов. На ленте T_2 группы будут разделены (последний символ каждой группы будет особым), а на ленте T_3 группы не будут разделены. Длина записи на каждой из лент будет равна $2^i - 1$.

(в) Посредством ленты T_3 машина определяет начальные 2^i i -блоков и записывает $2^i(i+1)$ символов i -базы входного слова на ленту T_4 . Одновременно машина записывает в унарной форме на ленте T_5 число $2^i(i+1)$.

(г) На ленте T_6 машина записывает копию (2^i+1) -го входного блока.

В течение первого этапа своей работы машина воспринимает входные символы на каждом шаге вычисления, и все это время выходным символом является 0. Первый этап завершается восприятием $((2^i+1)(i+1))$ -го входного символа (им является *, если все блоки имеют длину $i+1$); на этом шаге выходной символ не выдается.

Второй этап работы начинается с $((2^i+1)(i+1)+1)$ -го шага вычисления. В течение этого этапа машина не будет воспринимать какие-либо входные символы и не выдает никаких выходных символов.

(а) Лента T_5 теперь используется в качестве ленты устройства, вычисляющего функцию $U(n)$ в реальное время. В процессе работы этого устройства головка на ленте T_4 возвращается к началу расположенной на этой ленте записи (которое отмечено заранее). Устройство завершает вычисление $U(2^i(i+1))$ в реальное время (в течение $C_1 \cdot U(2^i(i+1))$ шагов) и записывает в унарной форме это число на ленте T_7 .

(б) Машина самое большее за $U(2^i(i+1))+1$ шагов вычисляет $\lceil \log U(2^i(i+1)) \rceil$ (используется построенная Ямада [2] машина, реализующая функцию $f(n)=2^n$), далее производится вычитание из этого числа i , и таким образом вычисляется $\lceil f(i) \rceil$. Число $\lceil f(i) \rceil$ в унарной форме записывается на ленте T_8 .

Третий этап начинается после окончания второго этапа. На третьем этапе машина не воспринимает каких-либо входных символов и не выдает выходных символов. На ленту T_9 машина последовательно заносит в закодированном виде 2^i i -блоков, которые записаны на ленте T_4 . Одновременно машина проверяет, удовлетворяют ли i -блоки второму условию из определения A_U (имеются в виду i -блоки слов из A_U). Головка на ленте

T_9 , сдвигаясь вправо начиная с отмеченного квадрата, заносит на эту ленту в закодированном виде блок с ленты T_4 , именно блоку $u_1 u_2 \dots u_i *$ ставится в соответствие квадрат ленты (этот квадрат отмечается), на который она приходит, осуществив сдвиг на

$$\sum_{k=1}^i u_k \cdot 2^{k-1}$$

квадратов вправо.

Для i -блока слова из A_U сдвиг производится максимум на

$$\sum_{k=1}^{[f(i)]} 2^{k-1}$$

квадратов и этот сдвиг соответствует блоку

$$\underbrace{11 \dots 1}_{[f(i)]} \underbrace{00 \dots 0}_{i - [f(i)]} *.$$

Отсутствие сдвига (т. е. начальный квадрат) соответствует i -блоку $\underbrace{00 \dots 0}_i *$.

После окончания кодирования i -блока головка на ленте T_9 возвращается к отмеченному начальному квадрату и начинается кодирование следующего i -блока, пока не будет исчерпана запись на ленте T_4 . При реализации сдвигов на 2^{k-1} ($k = 1, 2, \dots, [f(i)]$) квадратов вправо используются группы, записанные на ленте T_2 . Для записи кода произвольного i -блока слова из A_U требуется число квадратов, не превосходящее $2^{[f(i)]} < 2^{f(i)} = U(2^i(i+1))/2^i$, и кодирование является взаимно однозначным. При кодировании i -блока слова из A_U машина совершает самое большее $2^{f(i)+1}$ шагов, так как головка на ленте T_9 всегда возвращается к отмеченному начальному квадрату.

Рассмотрим теперь *четвертый этап*. Машина, после того как она закодировала базу (первые 2^i i -блоков), проверяет $(2^i + 1)$ -й блок (т. е. проверяет, совпадает ли он с каким-либо блоком из базы). На ленте T_9 головка движется подобно тому, как она двигалась, кодируя блок, но соответствующий квадрат не отмечается (как на третьем этапе), а проверяется, отмечен он или нет, затем головка возвращается к начальному квадрату и машина выдает выходной символ 1 (или 0).

Пятый этап. Машина воспринимает входные символы следующего блока, и они проверяются головкой на ленте T_9 так же, как на четвертом этапе. Если u_k есть k -й ($k \leq [f(i)]$) символ блока, то головка движется на $u_k \cdot 2^{k-1}$ квадратов вправо и машина выдает выходной символ 0. Затем машина воспринимает $(k+1)$ -й символ рассматриваемого i -блока и т. д. На том шаге,

когда входным символом является *, машина определяет, является ли квадрат под головкой на ленте T_9 отмеченным, затем эта головка возвращается к начальному квадрату и машина выдает выходной символ 1 или 0. Машина работает так до тех пор, пока не будет достигнута максимальная возможная длина входного слова, самое большее $2^{i+1}(i+1)$ символов, т. е. самое большее 2^{i+1} i -блоков, затем машина выдает только нули (в соответствии с определением A_U). Число блоков, следующих после $(2^i + 1)$ -го блока, контролируется с помощью ленты T_3 .

Оценка временной функции машины M_U . Мы хотим произвести оценку временной функции $\tau_{A_U}^{M_U}(n)$. Пусть длина первого блока входного слова v длины n будет $i+1$. Тогда могут иметь место три случая:

- (1) $n < (2^i + 1)(i + 1)$,
- (2) $(2^i + 1)(i + 1) \leq n \leq 2^{i+1}(i + 1)$,
- (3) $n > 2^{i+1}(i + 1)$.

В первом случае машина M_U обрабатывает слово v за n шагов (смотри первый этап работы машины M_U). Во втором случае время работы будет:

$$\begin{aligned}
 t(v) &\leq (2^i + 1)(i + 1) + && \text{— запись } 2^i + 1 \text{ } i\text{-блоков на ленты } T_4 \text{ и } T_6 \text{ (первый этап работы);} \\
 &+ C_1 \cdot U(2^i(i + 1)) + && \text{— запись } U(2^i(i + 1)) \text{ символов на ленту } T_7 \text{ (второй этап (а));} \\
 &+ U(2^i(i + 1)) + && \text{— „вычисление“ } [f(i)] \text{ и запись } [f(i)] \\
 &+ i + 1 + && \text{на ленту } T_8 \text{ (второй этап (б));} \\
 &+ 2^i \cdot 2 \cdot \frac{U(2^i(i + 1))}{2^i} + && \text{— кодирование базы (третий этап);} \\
 &+ 2^i \cdot 2 \cdot \frac{U(2^i(i + 1))}{2^i} + && \text{— проверка самое большее } 2^i \text{ } i\text{-блоков (четвертый и пятый этапы);}
 \end{aligned}$$

откуда окончательно

$$t(v) \leq (2^i + 2) \cdot (i + 1) + (C_1 + 5) \cdot U(2^i(i + 1)),$$

из чего следует (так как $n \leq \frac{U(n)}{2}$), что $t(v) \leq C \cdot U(2^i(i + 1))$,

где константа C зависит только от функции $U(n)$ и способа ее вычисления. Так как временная функция $U(n)$ монотонно возрастающая, то $t(v) \leq C \cdot U(n)$.

В третьем случае входное слово v может быть представлено как $v_1 v_2$, где длина слова v_1 равна $2^{i+1}(i + 1)$. Время обработки слова v будет

$$\begin{aligned}
 t(v) &= t(v_1) + (n - 2^{i+1}(i + 1)) \leq C \cdot U(2^{i+1}(i + 1)) + \\
 &\quad + n - 2^{i+1}(i + 1) < C \cdot U(n).
 \end{aligned}$$

(Последнее неравенство имеет место, так как функция $U(n)$ монотонно возрастающая и $C > 1$.)

Рассмотрев все три случая, мы имеем

$$\tau_{AU}^{MU}(n) \leq C \cdot U(n), \quad \text{ч. т. д.}$$

2. ПРЕДЕЛЬНАЯ ТЕОРЕМА

Предварительно мы докажем следующую теорему.

Теорема 2. Если машина Тьюринга M со входом распознает множество A_U , то для бесконечно многих значений n имеет место следующее неравенство:

$$\tau_{AU}^M(n) > C \cdot U(n),$$

где $C > 0$ — константа, зависящая только от числа лент и числа символов внутреннего алфавита машины M .

Доказательство подобно доказательству для множества A у Хенни [1]. Мы дадим только его краткий набросок.

Будем считать две i -базы «различимыми» в том и только в том случае, когда существует i -блок, который является частью одной i -базы и не является частью другой i -базы. Можно выбрать $2^{[f(i)]} - 1$ попарно различных i -баз, состоящих из i -блоков и удовлетворяющих условиям, наложенным на множество A_U . Обозначим через a_j ($j = 1, 2, \dots, 2^{[f(i)]} - 1$) различные i -базы, которые соответствуют какому-то так определенному выбору i -баз. Далее пусть b_j ($j = 1, 2, \dots, 2^{[f(i)]} - 1$) — произвольные i -слова (напомним, что i -слово — это слово, содержащее конечное число $k = 0, 1, 2, \dots$, i -блоков), которые состоят самое большее из $2^i - 1$ i -блоков, удовлетворяющих условиям, наложенным на i -блок множества A_U . Тогда имеет место следующая

Лемма. Для каждого $i \geq 1$ существуют положительное целое j ($1 \leq j < 2^{[f(i)]}$) и i -блок d , такие, что всякой распознающей множество A_U машине Тьюринга для обработки слова $a_j b_j d$ требуется время (число шагов), большее чем v_i , где

$$v_i = \frac{2^{[f(i)]} - \log Q - t \cdot \log s - 1}{2 \cdot t \cdot \log s}.$$

Здесь Q — число внутренних состояний, t — число лент, а s — число символов внутреннего алфавита рассматриваемой машины.

Доказательство.

(а) Пусть $Q \cdot s^t > 2^{2^{[f(i)]}-1}$, тогда $\log Q + t \cdot \log s > 2^{[f(i)]} - 1$, поэтому $2^{[f(i)]} - \log Q - t \cdot \log s - 1 < 0$, откуда следует, что

$$\frac{2^{[f(i)]} - \log Q - t \cdot \log s - 1}{2 \cdot t \cdot \log s} < 0.$$

(б) Пусть $Q \cdot s^t \leq 2^{2^{[f(i)]}-1}$. Введем понятие x -конфигурации машины Тьюринга. В x -конфигурацию входят состояние машины, символы, расположенные на лентах под головками, и x символов слева и справа от них (всего $2x + 1$ символов на каждой ленте). Таким образом, имеется самое большее $Q \cdot s^{t(2x+1)}$ различных x -конфигураций данной машины. Пусть x_i — наибольшее целое, удовлетворяющее соотношению

$$Q \cdot s^{t(2x_i+1)} \leq 2^{2^{[f(i)]}-1} < 2^{2^{[f(i)]}} - 1.$$

Разумеется, $x_i \geq 0$ согласно предположению (б). Произведя логарифмирование, получим

$$\log Q + 2x_i t \cdot \log s + t \cdot \log s \leq 2^{[f(i)]} - 1.$$

Из последнего неравенства и того факта, что x_i — наибольшее целое число j , удовлетворяющее неравенству $Qs^{t(2j+1)} \leq 2^{2^{[f(i)]}-1}$, следует, что

$$0 \leq x_i \leq \frac{2^{[f(i)]} - \log Q - t \cdot \log s - 1}{2 \cdot t \cdot \log s} < x_i + 1.$$

Так как $Q \cdot s^{t(2x_i+1)} < 2^{2^{[f(i)]}} - 1$, то число i -слов $a_j b_j$ (где $j = 1, 2, \dots, 2^{[f(i)]} - 1$) больше чем число различных x_i -конфигураций рассматриваемой машины. Поэтому существуют два i -слова $a_p b_p$ и $a_q b_q$ (где $p \neq q$), такие, что если эти слова подать на вход машины, то в обоих случаях x_i -конфигурации в моменты времени, когда выдается последний выходной символ, будут равны. Но базы a_p, a_q различимы; следовательно, существует i -блок, который является частью одной, но не является частью другой базы. Обозначим его буквой d . Предположим, что i -блок d обрабатывается меньше чем за $x_i + 1$ шагов, т. е. в процессе обработки блока d головки машины не покидают квадратов, входящих в начальную x_i -конфигурацию. Машина обрабатывает таким образом последний i -блок d входных слов $a_p b_p d, a_q b_q d$. Итак, либо оба входных слова $a_p b_p d, a_q b_q d$ воспринимаются машиной, либо оба отвергаются.

Мы пришли к противоречию, так как i -блок является частью одной из баз a_p, a_q , но не является частью другой. Таким образом, время обработки i -блока d входного слова $a_p b_p d$

(и также соответственно время обработки i -блока d входного слова $a_q b_q d$) равно или больше чем $x_i + 1 > v_i$.

Следствие. Если a_j ($j = 1, 2 \dots 2^{[f(i)]} - 1$) — различные i -базы, то существуют положительное целое k и i -слово h , такие, что

(1) i -слово h содержит 2^i i -блоков, удовлетворяющих условиям, наложенным на i -блоки множества A_U ;

(2) при обработке машиной Тьюринга слова $a_k h$ время обработки каждого из 2^i i -блоков слова h больше чем v_i .

Доказательство. Пусть имеются различные i -базы a_j , тогда в соответствии с леммой существуют положительные целые p, q и i -блок d_1 , такие, что при переработке слов $a_p d_1, a_q d_1$ время обработки последнего i -блока (d_1) больше чем v_i . Возьмем i -слова a'_j ($j = 1, 2 \dots 2^{[f(i)]} - 1$), такие, что $a'_p = a_p d_1, a'_q = a_q d_1$ и $a'_j = a_j$ для остальных j , и применим к ним лемму. Мы получим снова то же самое число i -слов a''_j и т. д. Этот процесс закончится, когда будет получена группа i -слов, по крайней мере одно из которых имеет длину $2^{i+1}(i+1)$, т. е. содержит 2^{i+1} i -блоков. Ясно, что такое i -слово обладает указанными в формулировке следствия свойствами.

Теперь мы перейдем к доказательству теоремы 2.

Введем следующее обозначение: $n_j = 2^{j+1}(j+1)$. Для каждого положительного целого n существует i , такое, что

$$n_i = 2^{i+1}(i+1) \leq n < 2^{i+2}(i+2) = n_{i+1}.$$

Тогда $\tau(n) \geq \tau(n_i)$, и в соответствии со следствием

$$\tau(n_i) > 2^i v_i = \frac{2^i (2^{[f(i)]} - \log Q - t \cdot \log s - 1)}{2 \cdot t \cdot \log s},$$

где Q, t, s — константы; поэтому для больших значений i имеет место неравенство

$$\tau(n) > \frac{2^{i+[f(i)]}}{3 \cdot t \cdot \log s} > \frac{2^{i+f(i)}}{6 \cdot t \cdot \log s},$$

так как $2^{[f(i)]} > 2^{f(i)-1}$. Далее, так как $2^{f(i)} = U(2^i(i+1))/2^i$ (смотри определение множества A_U), имеет место следующее неравенство:

$$\tau(n) > \frac{U(2^i(i+1))}{6 \cdot t \cdot \log s}.$$

Для $i > 3$ выполняется неравенство $2^i(i+1) > 2^{i+2}(i+2)/5$ и, поскольку функция U монотонно возрастает, имеет место неравенство

$$\tau(n) > \frac{U(n_{i+1}/5)}{6 \cdot t \cdot \log s},$$

откуда следует, что

$$\tau(n) > \frac{U(n/5)}{6 \cdot t \cdot \log s},$$

поскольку $n_{i+1} > n$.

Для завершения доказательства теоремы 2 достаточно показать, что для бесконечно многих значений n имеет место неравенство $U(n/5) > U(n)/25$. Предположим, что это высказывание ложно, т. е. что для почти всех n выполняется неравенство $U(n/5) \leq U(n)/25$ или $25 \cdot U(n/5) \leq U(n)$.

Тогда можно выбрать такое n_0 , что

$$\begin{aligned} U(n_0) &> 0, \\ U(5n_0) &\geq 25U(n_0), \\ &\vdots \\ U(5^t n_0) &\geq 5^{2t} \cdot U(n_0). \end{aligned}$$

Согласно предположению, $U(n) \leq n^{2-\varepsilon}$, поэтому

$$\limsup_{n \rightarrow \infty} U(n)/n^{2-\varepsilon} < \infty.$$

Таким образом, последовательность

$$\frac{U(5^t \cdot n_0)}{(5^t \cdot n_0)^{2-\varepsilon}} \quad (\text{для } t = 0, 1, 2, \dots)$$

имеет $\limsup_{n \rightarrow \infty}$, меньший, чем ∞ , но

$$\frac{U(5^t \cdot n_0)}{(5^t \cdot n_0)^{2-\varepsilon}} \geq \frac{5^{2t} U(n_0)}{5^{2t} n_0^{2-\varepsilon}} \cdot 5^{\varepsilon t} \rightarrow \infty,$$

что противоречит предположению.

Подведем теперь итог полученным результатам. Каждой монотонно возрастающей функции $U(n)$, которая вычислима в реальное время и для которой имеет место соотношение $n \leq U(n) \leq n^{2-\varepsilon}$, можно поставить в соответствие множество

слов $A_U \subseteq A$, такое, что

(1) A_U распознается машиной Тьюринга со входом M_U за время $T(n) = C_1 \cdot U(n)$ (согласно теореме 1);

(2) если машина Тьюринга со входом распознает A_U , тогда для бесконечно многих значений n ее временная функция $\tau^M(n) > C_M \cdot U(n)$, где C_M — константа, зависящая от числа лент и от числа символов во внутреннем алфавите машины M (согласно теореме 2).

Непосредственным следствием двух предыдущих теорем является следующая:

Предельная теорема. Пусть T, U — монотонно возрастающие временные функции, и пусть функция U вычислима в реальное время, причем $n \leq \frac{1}{2} U(n) \leq n^2 - \varepsilon$ для подходящего $\varepsilon > 0$. Далее пусть $\lim_{n \rightarrow \infty} T(n)/U(n) = 0$. Тогда можно построить множество, которое распознается за время $U(n)$, но не распознается за время $T(n)$.

Доказательство. Примером такого множества является множество A_U .

ЛИТЕРАТУРА

1. Hennie F. C., On-line Turing machine computations, *IEEE Trans.*, EC-15 (1966), 35—44. (См. стр. 249—270 настоящего сборника.)
2. Yamada H., Real-time computation and recursive functions not real-time computable, *IRE Trans.*, EC-11 (1962), 753—760. (См. стр. 139—155 настоящего сборника.)

СЛОЖНОСТЬ ВЫЧИСЛЕНИЙ НА ОДНОЛЕНТОЧНЫХ МАШИНАХ ТЬЮРИНГА ¹⁾

Дж. Хартманис

ПРЕДВАРИТЕЛЬНЫЕ ЗАМЕЧАНИЯ

На протяжении всей статьи мы имеем дело с количественными аспектами вычислений на одноленточных машинах Тьюринга с записью на ленте. Мы предположим, что машина Тьюринга используется для распознавания последовательностей в некотором конечном алфавите I .

Множество всех конечных последовательностей (цепочек) в алфавите I мы обозначим через I^* , а длину последовательности $w = x_1 x_2 \dots x_k$, $x_i \in I$, $1 \leq i \leq k$ обозначим через $l(w)$ (в данном случае $l(w) = k$).

Пустая последовательность обозначена через Λ и $l(\Lambda) = 0$. Лента машины Тьюринга M неограничена вправо, и входная цепочка $w = x_1 x_2 \dots x_k$ (в I^*) записана на первых k ячейках ленты, а остальная часть ленты в начале вычислений пуста. Первая ячейка ленты, кроме x_1 , содержит единственный маркер (скажем, штрих при символе x_1), который сохраняется на протяжении всего вычисления, чтобы предотвратить уход машины с ленты. Следуя Хенни [1], мы употребляем слово **отрезок (сегмент) ленты** для обозначения конечной части ленты, а слово «лента» используется для обозначения бесконечной последовательности ячеек, в которых могут записываться символы. Тогда равенство $t = ww_1$ показывает, что лента t получена сочленением отрезка ленты w с бесконечной вправо лентой w_1 . Чистая лента (неограниченная вправо) обозначается через w_b .

Машина Тьюринга всегда начинает работу с того, что ее читающая головка обозревает крайнюю слева ячейку ленты (первый символ входной цепочки). При каждой операции машина M обозревает ячейку ленты, находящуюся под читающей головкой; считываемый с ленты символ и настоящее состояние машины однозначно определяют:

(1) символ (из конечного алфавита), который печатается в обозреваемой головкой ячейке,

¹⁾ Hartmanis J., Computational complexity of one-tape Turing machine computations, *J. Assoc. Comput. Mach.*, 15, № 2 (1968), 325—339.

2) новое состояние (из конечного множества состояний), в которое переходит M , и

3) движение читающей головки, которая двигается на одну ячейку вправо или влево по ленте, вплоть до момента останова машины при ее переходе в *отвергающее* или *допускающее* состояние.

Множество конечных последовательностей A в I , $A \subseteq I^*$, *допускается* или *распознается* машиной Тьюринга M тогда и только тогда, когда M останавливается при любом входе w из I^* , и допускает вход w , если он принадлежит A , и отвергает его в противном случае, переходя в допускающее или отвергающее состояние соответственно.

Ниже мы определим три количественные характеристики сложности вычислений на одноленточных машинах Тьюринга со входом.

1. Число операций, совершаемых машиной Тьюринга в процессе выяснения того, допускается или отвергается входная цепочка, является нашей мерой времени. Пусть $T(n)$ является вычислимой функцией, отображающей неотрицательные целые числа в неотрицательные целые числа, $T: \mathcal{U} \rightarrow \mathcal{U}$. Тогда мы скажем, что множество A является **$T(n)$ -распознаваемым** тогда и только тогда, когда существует машина Тьюринга M , которая допускает A и которая совершает над каждым входом длины n не более чем $T(n)$ операций.

2. Объем ленты, используемый машиной Тьюринга, является нашей мерой памяти. Пусть $L(n)$ — вычислимая функция, $L: \mathcal{U} \rightarrow \mathcal{U}$. Тогда мы будем говорить, что множество A **$L(n)$ -распознаваемо** или **распознаваемо на $L(n)$ -ленте** тогда и только тогда, когда существует машина Тьюринга, допускающая A и использующая для каждого входа длины n не более чем $L(n)$ ячеек ленты.

3. Последняя мера сложности основана на числе моментов времени, в которые M пересекает границу между двумя ячейками. Пусть $R(n)$ является вычислимой функцией, $R: \mathcal{U} \rightarrow \mathcal{U}$. Тогда множество A называется **$R(n)$ -распознаваемым**, если существует машина Тьюринга M , допускающая A , которая работает над каждым входом длины n , не пересекая каждую границу между соседними ячейками ленты более чем $R(n)$ раз¹⁾. (См. связанные с этим работы [1—4].)

Скажем, что машина Тьюринга M *определяет* $T(n)$, если для всякого входа длины n машина использует не более чем $T(n)$

¹⁾ В работе [4] функция $R(n)$ называется режимом работы машины Тьюринга. — Прим. ред.

операций, а для некоторого входа длины n — в точности $T(n)$ операций.

Аналогично машина M *определяет* $L(n)$ или $R(n)$, если M никогда не использует более чем $L(n)$ ячеек ленты или $R(n)$ пересечений границы между ячейками ленты для входов длины n , а для некоторого входа длины n использует в точности $L(n)$ ячеек ленты или совершает ровно $R(n)$ пересечений некоторой границы.

Теперь мы дадим краткое описание текущих последовательностей («следов»), являющихся основным аналитическим методом этой статьи. Этот метод рассматривался сначала в [4], а затем в [1]. Для машины Тьюринга M и ленты t с каждой границей между двумя соседними ячейками ленты t мы свяжем упорядоченную последовательность состояний $s(1), s(2), \dots, s(n)$; $s(i)$ есть состояние, в котором машина M в i -й раз пересекает эту границу в процессе вычисления; назовем эту последовательность *следом*¹⁾. Если $t = w_1 w_2$, то след на границе между w_1 и w_2 мы обозначим через $C(w_1; w_2)$.

Ясно, что след $C(w_1; w_2)$ полностью описывает информацию, переносимую через границу между w_1 и w_2 машиной M в процессе вычисления. Таким образом, мы легко устанавливаем следующий результат, утверждающий, что любой сегмент ленты между одинаковыми следами можно выбросить без изменения вычислений на оставшейся части ленты [1].

Лемма 1. Если $t = w_1 w_2 w_3 w_4 w_5$ и для M $C(w_1 w_2; w_3 w_4 w_5) = C(w_1 w_2 w_3; w_4 w_5)$, то $C(w_1; w_2 w_4 w_5) = C(w_1; w_2 w_3 w_4 w_5)$ и $C(w_1 w_2 w_4; w_5) = C(w_1 w_2 w_3 w_4; w_5)$.

Заметим, что машина M изменяет свое состояние перед тем, как она сдвигается, и поэтому, если M допускает (отвергает) $t = w_1 w_2 w_3 w_4 w_5$, она допустит (отвергнет) $t = w_1 w_2 w_4 w_5$.

РАСПОЗНАВАНИЕ НЕРЕГУЛЯРНЫХ МНОЖЕСТВ

Известно из [5], что если множество A распознается на одноленточной машине Тьюринга с записью на ленте, которая ничего не пишет на своей ленте, то A является регулярным множеством и может распознаваться конечным автоматом или, что эквивалентно этому, машиной Тьюринга, которая обзорекает каждый входной символ только один раз.

¹⁾ Термин взят из работы [4]. — Прим. перев.

Недавно Хенни [1] обобщил этот результат и показал, что если M допускает A и для некоторой константы k $R(n) \leq k$ или $T(n) \leq kn$, то множество A регулярно¹⁾.

Теперь мы покажем, что существуют точные нижние границы скорости роста $R(n)$ и $T(n)$, при которых возможно распознавание нерегулярных множеств на одноленточных машинах Тьюринга. Первые две теоремы были получены Трахтенбротом [4] и независимо автором.

Теорема 1. Если A $R(n)$ -распознаваемо и $\lim_{n \rightarrow \infty} \frac{R(n)}{\log n} = 0$, то A является регулярным множеством.

Доказательство. Мы покажем, что из $\lim_{n \rightarrow \infty} \frac{R(n)}{\log n} = 0$ следует, что $R(n) \leq k$ для некоторой постоянной k , а стало быть, и результат Хенни о том, что A регулярно.

Пусть M распознает A и определяет $R(n)$. (Таким образом, $R(n)$ является самым длинным следом, порождаемым машиной на входах длины n .) Если $R(n)$ неограничена, то существует бесконечная последовательность целых чисел $0 < n_1 < n_2 < \dots$, такая, что $R(n_i) > R(n)$ для $n_i > n$. Итак, M порождает след длины $R(n_i)$ в первый раз при входном слове длины n_i .

Мы теперь покажем, что для входных слов длины n_i , на которых возникает след длины $R(n_i)$, никакой след не может появиться более чем дважды. Пусть $t_i = w_1 w_2 w_3 w_4$, $w_2, w_3 \neq \Lambda$, и пусть M на $t = w_1 w_2 w_3 w_4 w_b$ порождает следы

$$C(w_1; w_2 w_3 w_4 w_b) = C(w_1 w_2; w_3 w_4 w_b) = C(w_1 w_2 w_3; w_4 w_b).$$

Тогда по предыдущей лемме мы знаем, что, когда M начинает работу с входным словом $t' = w_1 w_3 w_4 w_b$ или $t'' = w_1 w_2 w_4 w_b$, она порождает следы длины $R(n_i)$ на $w_1 w_3 w_4 w_b$ или на $w_1 w_2 w_4 w_b$. Но, поскольку эти сегменты короче чем t_i , мы приходим к противоречию с тем фактом, что след длины $R(n_i)$ достигается впервые на ленте длины n_i .

Таким образом, на последовательности входных сегментов $t_1, t_2, \dots$ длин $n_1 < n_2 < \dots$ соответственно никакой след не порождается более чем два раза. Мы используем этот факт для вычисления возможной скорости роста $R(n_i)$.

Если у M есть Q состояний ($Q \geq 2$), то число различных следов длины не более r дается формулой

$$\sum_{i=0}^r Q^i = \frac{Q^{r+1} - 1}{Q - 1} \leq Q^{r+1}$$

¹⁾ До Ф. Хенни этот (и даже более сильный) результат был получен в работе [4]. — Прим. перев.

Если на сегменте длины n_i никакой след не может быть порожден больше чем два раза, то $2Q^{R(n_i)+1} \geq n_i$. Логарифмируя обе части неравенства по основанию Q , получаем, что $R(n_i) + 1 + \log 2 \geq \log n_i$, откуда $R(n_i) \geq \log n_i - 2$. Но тогда

$$\lim_{n \rightarrow \infty} \frac{R(n)}{\log n} \neq 0.$$

Таким образом, равенство

$$\lim_{n \rightarrow \infty} \frac{R(n)}{\log n} = 0$$

влечет, что $R(n) \leq k$ и, стало быть, A регулярно.

Следствие 1. Если машина Тьюринга M распознает нерегулярное множество A и определяет $R(n)$, то

$$\sup_{n \rightarrow \infty} \frac{R(n)}{\log n} > 0.$$

Доказательство следует прямо из теоремы.

Дальше мы покажем, что существуют нерегулярные множества, распознаваемые с $R(n) = 2\lceil \log_2 n \rceil$. (Символ $\lceil x \rceil$ здесь обозначает наименьшее целое число k , такое, что $k \geq x$.)

Лемма 2. Существует одноленточная машина Тьюринга M , которая, начиная работу с ленты $t = 1^n \omega_b$, вычисляет с $R(n) = 2\lceil \log_2 n \rceil$ двоичное представление n (т. е. M останавливается, написав на ленте двоичное разложение числа n).

Доказательство. Машина M движется слева направо по сегменту и отмечает первую, третью, пятую и т. д. неотмеченные ячейки, затем возвращается и повторяет этот процесс до тех пор, пока все ячейки сегмента не будут отмечены. Очевидно, что в двоичном разложении

$$n = \sum_{i=0}^k a_i 2^i$$

коэффициент a_{i-1} есть 1, если крайняя правая неотмеченная ячейка отмечается на i -м колебании, в противном случае $a_{i-1} = 0$. Таким образом, чтобы вычислить $a_k a_{k-1} \dots a_0$, машина просто пишет a_i в соответствии с указанной процедурой. Поскольку этот процесс выполняется за $\lceil \log_2 n \rceil$ колебаний, мы видим, что существует машина M , вычисляющая двоичное разложение с $R(n) = 2\lceil \log_2 n \rceil$ ¹⁾.

¹⁾ Если отмечать ячейки и при обратном движении головки, то требуемую работу можно выполнить с $R(n) = \lceil \log_2 n \rceil + 2$. — Прим. ред.

Из приведенного доказательства следует (как было уже показано в [1]), что $n = 2^k$ тогда и только тогда, когда процесс оканчивается за k колебаний и крайняя правая ячейка отмечается в последнем колебании. Таким образом, нерегулярное множество $\{1^{2^k} | k = 1, 2, \dots\}$ является $(R(n) = 2)\log_2 n$ -распознаваемым.

Дальше мы покажем, что существует точная достижимая нижняя временная граница при вычислении нерегулярных предикатов. Этот результат и следствие из него были впервые получены Трахтенбротом [4] и независимо от него автором¹⁾.

Теорема 2. Если A является $T(n)$ -распознаваемым и $\lim_{n \rightarrow \infty} \frac{T(n)}{n \log n} = 0$, то A регулярно.

Доказательство. Заметим, что время вычислений равно сумме длин всех следов в вычислении. Дальше доказательство теоремы 1 показывает, что коль скоро множество A нерегулярно, то существует бесконечно много входных сегментов $t_1, t_2, t_3, \dots$ длин $n_1 < n_2 < n_3 < \dots$, на которых ни один след не порождается более чем дважды. В этом доказательстве было показано далее, что, даже если M порождает кратчайшие следы на сегментах t_i , длина самого большого следа $R(n_i)$ должна быть такой, что $R(n_i) \geq \log_Q n_i - 2$.

Таким образом, для всех i время вычислений $T(n_i)$ на t_i должно быть таким, что

$$T(n_i) \geq 2 \sum_{j=0}^r jQ^j; \quad r = \lfloor \log_Q n_i \rfloor - 3,$$

поскольку существует Q^j различных следов длины j и ни один след не может встретиться чаще чем два раза. Заметим, что машина M не обязана порождать все следы длины, меньшей чем $\log_Q(n_i) - 2$, но при замене коротких следов на более длинные неравенство может только усилиться. Возвращаясь к неравенству, мы видим, что

$$T(n_i) \geq 2 \sum_{j=0}^r jQ^j \geq 2rQ^r \geq 2(\log_Q n_i - 3)n_iQ^{-3},$$

и поэтому

$$\lim_{n \rightarrow \infty} \frac{T(n)}{n \log n} \neq 0.$$

Таким образом, из того, что $\lim_{n \rightarrow \infty} \frac{T(n)}{n \log n} = 0$, следует, что A регулярно.

¹⁾ Следует отметить, что автором этот результат здесь публикуется впервые, а работа [4] опубликована в 1964 г. — Прим перев.

Следствие 2. Если M распознает нерегулярное множество и определяет $T(n)$, то

$$\sup_{n \rightarrow \infty} \frac{T(n)}{n \log n} > 0.$$

Доказательство немедленно следует из теоремы.

Поскольку множество $A = \{1^{2^k} \mid k = 1, 2, \dots\}$ может быть распознано с $\lceil \log n \rceil$ колебаниями для входов длины n , мы видим, что A $(T(n) = 2n \lceil \log n \rceil)$ -распознаваемо.

Предыдущий результат можно расширить, чтобы получить соотношение между временем и объемом памяти, используемыми при вычислении с неограниченной длиной следов.

Теорема 3. Пусть M останавливается для всех входов и определяет $L(n)$, $R(n)$ и $T(n)$. Тогда $\lim_{n \rightarrow \infty} \frac{n}{L(n)} = 0$ влечет существование таких $C > 0$ и N , что

$$R(n) \geq C \log L(n),$$

$$T(n) \geq CL(n) \log L(n) \quad \text{для } n \geq N.$$

Доказательство. Сначала мы покажем, что для каждой входной ленты $t = \omega \omega_b$ M не может породить два одинаковых следа на первоначально пустой ленте ω_b . Если $C(\omega b^r; b^k \omega_b) = C(\omega b^r b^k; \omega_b)$, то, согласно лемме 1, мы можем удалить сегмент b^k и при этом не изменить вычисления, исключая то, что будет на k следов меньше. С другой стороны, удаление b^k из ω_b оставляет ленту ω_b неизменной, и поэтому $t = \omega \omega_b$ не изменяется. Таким образом, M будет порождать точно столько же следов, как и прежде, и (ввиду того что M остановится) мы заключаем, что $k = 0$.

Поскольку

$$\lim_{n \rightarrow \infty} \frac{n}{L(n)} = 0,$$

существует целое число N_1 , такое, что $L(n) \geq 2n$ для $n \geq N_1$. Следовательно, для каждого n , $n \geq N_1$, существует вход длины n , для которого $L(n) - n \geq \frac{1}{2}L(n)$, и мы видим, что при этом входе M использует $\frac{1}{2}L(n)$ ячеек ленты ω_b . Отсюда следует, что M должна порождать по крайней мере $\frac{1}{2}L(n)$ различных следов, поскольку на ω_b никакие два следа не повторяются. Существует

$$\sum_{i=0}^r Q^i \leq Q^{r+1}$$

различных следов длины r или меньше ($Q \geq 2$). Для того чтобы породить $1/2 L(n)$ различных следов, нужно иметь $Q^{R(n)+1} \geq 1/2 L(n)$, и поэтому

$$R(n) \geq \log_Q L(n) - 1 - \log_Q 2.$$

Отсюда следует существование положительных констант C_1 и N_2 , таких, что

$$R(n) \geq C_1 \log_Q L(n) \quad \text{для } n \geq N_2.$$

Время вычисления $T(n)$, $n \geq N_1$, не меньше чем сумма длин $1/2 L(n)$ различных следов, порожденных M . Таким образом,

$$T(n) \geq \sum_{i=0}^r i Q^i, \quad r = \lfloor \log_Q L(n) \rfloor - 3,$$

и поэтому

$$T(n) \geq (\log_Q L(n) - 3) L(n) Q^{-3}.$$

Отсюда следует, что существуют две положительные константы C_2 и N_3 , такие, что

$$T(n) \geq C_2 L(n) \log_Q L(n) \quad \text{для } n \geq N_3.$$

Для получения требуемых неравенств положим $C = \min(C_1, C_2)$, а $N = \max(N_2, N_3)$.

Теорема 4. Пусть M останавливается при всех входах, определяет $R(n)$, $L(n)$, $T(n)$, и пусть функция $R(n)$ неограничена. Тогда существуют две положительные константы C_1 и C_2 , такие, что для бесконечно многих значений n_i

$$\begin{aligned} R(n_i) &\geq C_1 \log_Q L(n_i), \\ T(n_i) &\geq C_2 L(n_i) \log_Q L(n_i). \end{aligned}$$

Доказательство. Поскольку $R(n)$ — неограниченная функция, то существует бесконечно много значений $L(n_1) < L(n_2) < L(n_3) < \dots$, таких, что $R(n) < R(n_i)$, если $L(n) < L(n_i)$. Аргументами, подобными тем, которые были использованы при доказательствах теорем 1 и 2, можно показать, что ни один след не может повторяться больше чем два раза при вычислении, когда $R(n_i)$ достигается в первый раз (т. е. на кратчайшем $L(n)$). Снова вычисляя число разных следов, мы заключаем, что для некоторых $C_1 > 0$ и $N_1 > 0$

$$R(n_i) \geq C_1 \log_Q L(n_i) \quad \text{для } n_i \geq N_1.$$

Аналогично теперь мы можем подсчитать число операций, выполняемых в этих вычислениях, и установить, что

$$T(n_i) \geq C_2 L(n_i) \log_Q L(n_i) \quad \text{для } n_i \geq N_2.$$

Беря $n_i \geq \max(N_1, N_2)$, мы получаем искомое бесконечное множество целых положительных чисел, для которых выполняются упомянутые неравенства.

Предыдущие результаты показали, что существует резкий скачок времени вычислений, когда мы переходим от регулярных вычислений к нерегулярным. Далее мы возвратимся к классификации сложности нерегулярных множеств по их времени вычислений на одноленточных машинах Тьюринга. Связанные с этим результаты для многоленточных машин Тьюринга можно найти в [2, 6].

ИЕРАРХИИ ВЫЧИСЛЕНИЙ С ОГРАНИЧЕННЫМ ВРЕМЕНЕМ

В этом разделе мы исследуем классификацию нерегулярных множеств по времени, которое требуется для их распознавания.

Для вычислимой функции $T(n)$ ($R(n)$) назовем класс $T(n)$ -распознаваемых ($R(n)$ -распознаваемых) множеств последовательностей **классом сложности** и обозначим его через C_T (C_R). Следующий результат показывает, что существует бесконечно много классов сложности.

Лемма 3. *Если $T(n)$ — вычислимая функция, то существует рекурсивное множество последовательностей A , не принадлежащее C_T .*

Доказательство проводится обычным диагональным методом [2].

Далее мы покажем, что каждое вычисление можно ускорить в постоянное число раз, если допустить обычное сгущение входных цепочек, т. е. мы позволяем записывать несколько входных символов в одной ячейке ленты.

Теорема 5. *Если A $T(n)$ -распознаваемо, то A $(\lceil 1/2 T(n) \rceil)$ -распознаваемо.*

Доказательство. Пусть входное слово уплотнено на два символа в каждой ячейке ленты и A распознаваемо машиной M за время $T(n)$. Тогда с помощью метода, аналогичного тому, который был использован в доказательстве теоремы 2 в [2], мы можем показать, что существует машина M' , которая распознает A и совершает одну операцию для каждой двух операций M . Таким образом, M' распознает A за время $T'(n) = \lceil 1/2 T(n) \rceil$.

Следующий результат показывает, что для медленно растущих функций небольшое (нелинейное) возрастание времени вычисления достаточно для распознавания более сложных множеств. Для доказательства этого факта мы введем функции колебаний, которые очень легко вычисляются. Они используются

для вычисления числа колебаний над входным сегментом, совершаемых машиной M , и для окончания вычислений, если число колебаний станет слишком большим. Функции колебаний играют роль, аналогичную функциям, вычислимым в реальное время, в изучении времени вычислений на многоленточных машинах [2], и реализуемым (или «конструируемыми»¹⁾) функциям в изучении вычислений с ограниченной памятью [3].

Определение. Пусть F — монотонная возрастающая функция, отображающая множество натуральных чисел в себя, $F: \mathcal{N} \rightarrow \mathcal{N}$, такая, что для некоторого $Q > 0$ и для больших n $3n \leq F(n) \leq Q^n$. Тогда F является **функцией колебаний** в том и только в том случае, когда существует вычислимая монотонно возрастающая функция g , $g: \mathcal{N} \rightarrow \mathcal{N}$, такая, что множество

$$A = \{1^{g(k)} 0^{F(g(k)) - g(k)} \mid k = 1, 2, \dots\}$$

может распознаваться машиной Тьюринга M , совершающей не более чем $F^{-1}(n)$ колебаний над входным словом длины n .

Заметим, что $F(n)$ не может расти быстрее, чем экспонента, так как в противном случае

$$\sup_{n \rightarrow \infty} \frac{F^{-1}(n)}{\log n} = 0,$$

и по следствию 1 за $F^{-1}(n)$ колебаний могут определяться лишь регулярные множества. Нижняя граница для $F(n)$ явно используется в доказательстве теоремы 6 (см. также обсуждение после следствия 3).

Функции колебаний образуют интересный и богатый класс функций (который будет исследоваться далее). Здесь же нам достаточно отметить, что этот класс содержит много часто употребляемых функций.

Например,

$$F(k) = k^{p/q}, \quad p > q; \quad F(k) = 2^k; \quad F(k) = 2^{(p/q)k}, \quad p/q < 1;$$

$$F(k) = k \lceil \log_2 k \rceil^p, \quad p = 1, 2, \dots; \quad F(k) = k \lceil \log \log k \rceil$$

и т. д. Чтобы лучше понять функции колебаний, рассмотрим $F(k) = 2^k$. Возьмем $g(k) = k$. Тогда, используя лемму 2, мы видим, что множество $\{1^k 0^{2^k - k} \mid k = 1, 2, \dots\}$ является распознаваемым за k колебаний с $k \leq \log_2 n = F^{-1}(n)$. Таким образом, $F(k) = 2^k$ является функцией колебаний.

Чтобы понять, как используется вспомогательная функция $g(k)$, рассмотрим $F(k) = k^{p/q}$ (см. в [1] описание используемого метода. Пусть $g(k) = 2^{qk}$. Тогда для распознавания множе-

¹⁾ В нашей литературе такие функции называются также точными, емкостными и сигнализирующими. — *Прим. перев.*

ства $\{1^{2^{qk}}0^{2^{pk}-2^{qk}} \mid k=1, 2, \dots\}$ мы построим машину M , которая проверяет (с помощью процесса, описанного в лемме 2), будут ли длина входной последовательности и длина ее отрезка степенями двойки, и проверяет, будут ли числа колебаний, необходимых для выяснения этого вопроса для двух отрезков, находиться в отношении p/q . Это может быть сделано за kp колебаний над входной последовательностью. Поскольку $n=2^{pk}$, то $F^{-1}(n)=2^{kq} \geq kp$ (для больших n), и поэтому $F(k)=k^{p/q}$ является функцией колебаний.

Аналогичным методом можно показать, что много других функций являются функциями колебаний. Автор убежден в том, что функции колебаний следует изучить более детально и сравнить их свойства со свойствами функций, вычисляемых в реальное время, и реализуемых функций. Пока что это систематически еще не проводилось.

Теперь мы применим функции колебаний для обобщения результатов Хенни [1, теорема 4 и следствие 4] и покажем, что функции колебаний можно использовать для определения множеств последовательностей с точными оценками времени их распознавания.

Теорема 6. Если F является функцией колебаний, то существует множество последовательностей $[R(n)=F^{-1}(n)]$ - и $T(n)=nF^{-1}(n)$ -распознаваемых, но не $R_1(n)$ - или $T_1(n)$ -распознаваемых, если

$$\lim_{n \rightarrow \infty} \frac{R_1(n)}{F^{-1}(n)} = 0 \quad \text{или} \quad \lim_{n \rightarrow \infty} \frac{T_1(n)}{nF^{-1}(n)} = 0.$$

Доказательство. Покажем, что множество

$$A = \{w_i w w_i \mid w_i \in (0 \vee 1)^*, l(w_i) = g(k), w = a^{F[g(k)]-2g(k)}\}$$

удовлетворяет теореме. Для краткости положим $g(k)=k$. Поскольку $F(k)$ является функцией колебаний, мы можем за $k=l(w_i)$ колебаний узнать, будут ли удовлетворять условиям длины трех отрезков, и в то же время выяснить, одинаковы ли первый и третий отрезки. В этом вычислении $n=F(k)$ и поэтому $F^{-1}(n)=k$. Таким образом, множество A является $[R(n)=F^{-1}(n)]$ -распознаваемым. Дальше, так как каждое колебание не длиннее чем n , мы видим, что A распознаваемо за не более чем $T(n)=2nF^{-1}(n)$ операций. Таким образом (по теореме 5), A является $[T(n)=nF^{-1}(n)]$ -распознаваемым.

Дальше мы покажем, что если

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{nF^{-1}(n)} = 0,$$

то A не $T_1(n)$ -распознаваемо. Для этого заметим, что для фиксированного k существуют 2^k различных слов w_i , таких, что $w_i a^{F(k)-2k} w_i$ принадлежит A и $l(w_i) = k$. Положим

$$A_k = \{w_i a^{F(k)-2k} w_i \mid l(w_i) = k\} \subseteq A.$$

Тогда все следы, порожденные M в средних отрезках $w = a^{F(k)-2k}$ последовательностей A_k , должны быть различны, так как если $C(w_i a^p; a^q w_i) = C(w_j a^r; a^s w_j)$, $p \neq r$ или $w_i \neq w_j$, то M допускает (по [1, теорема 1]) последовательность $w_i a^p a^s w_j$, не принадлежащую A . Таким образом, при распознавании слов из A_k M должна использовать $2^k[F(k) - 2k]$ различных следов в средних отрезках w . Так как $F(k) \geq 3k$, то существует $C > 0$, такое, что $2^k[F(k) - 2k] \geq C 2^k F(k)$.

Существует $\sum_{i=0}^r Q^i \leq Q^{r+1}$ различных следов длины r или меньше для машины с Q состояниями, $Q \geq 2$. Для получения $C \cdot 2^k F(k)$ различных следов нужно иметь $r+1 \geq \log[C \cdot 2^k F(k)]$, и поэтому для некоторого $C_1 > 0$

$$r+1 \geq C_1 k + \log F(k).$$

Поскольку $n = F(k)$ и $k = F^{-1}(n)$, мы заключаем, что (для больших n) $r \geq C_1 \cdot F^{-1}(n)$, а, стало быть, $\sup_{n \rightarrow \infty} \frac{R(n)}{F^{-1}(n)} \neq 0$. Таким образом, A не будет $R_1(n)$ -распознаваемым, если

$$\lim_{n \rightarrow \infty} \frac{R_1(n)}{F^{-1}(n)} = 0.$$

Вычислим теперь нижнюю границу для времени вычислений. Если в вычислениях встречаются все следы, то граница времени вычислений для слов из A_k дается соотношением

$$\sum_{i=0}^r i Q^i / 2^k \geq \log[C 2^k F(k)] \cdot C \cdot 2^k \cdot F(k) / 2^k \geq C_3 F(k) k$$

при $C_3 > 0$. Снова ввиду $n = F(k)$ и $k = F^{-1}(n)$ мы получаем, что для некоторой последовательности в A_k время вычислений должно превосходить $C_3 n F^{-1}(n)$.

Таким образом, из

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{n F^{-1}(n)} = 0$$

следует, что множество A не $T_1(n)$ -распознаваемо.

Следствие 3. Пусть F является функцией колебаний. Тогда существует множество A , такое, что принадлежность A к $C_{nF^{-1}(n)}$ и к C_T влечет

$$\sup_{n \rightarrow \infty} \frac{T(n)}{nF^{-1}(n)} > 0,$$

а принадлежность A к $C_{F^{-1}(n)}$ и C_R влечет

$$\sup_{n \rightarrow \infty} \frac{R(n)}{F^{-1}(n)} > 0.$$

Доказательство немедленно следует из теоремы.

Приведенные выше результаты устанавливают для широкого класса функций (множество функций колебаний) множества последовательностей с хорошо определенным временем вычислений. К сожалению, рассуждение сохраняет силу только для медленно растущих временных функций, а именно для $T(n) \leq \leq n^2$. Легко видеть, что это ограничение, как во многих других аналогичных доказательствах, происходит оттого, что мы строим множества, в которых два отрезка длины n или менее должны проверяться на тождество. Это можно сделать в пределах n^2 операций, и, таким образом, эти результаты не удастся распространить сверх n^2 с помощью этих приемов.

Для получения результатов для более сложных вычислений, т. е. для больших временных функций, мы вынуждены применять диагональный метод. К сожалению, диагональное доказательство для одноленточных машин весьма затруднено, и результаты, полученные в этой статье для произвольно больших временных функций, значительно слабее, чем предыдущий. С другой стороны, есть предположение автора о том, что для произвольно больших временных функций $T(n)$ условие

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{T(n)} = 0$$

влечет $C_{T_1} \neq C_T$.

Следующий результат, полученный совместно с Дж. Хопкрофтом, является лучшим из полученных до сих пор.

Теорема 7. Пусть $T(n)$, $T(n) \geq n^2$, определяется машиной Тьюринга и вычисляется на $L(n) = \lfloor \log T(n) \rfloor$ -ленте. Тогда существует множество, являющееся $T(n) \lfloor \log T(n) \rfloor$ -определимым и не являющееся $T_1(n)$ -определимым для функции $T_1(n)$, такой, что

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{T(n)} = 0.$$

Доказательство. Мы дадим краткий набросок доказательства построением машины Тьюринга M , распознающей

за время $T(n)]\log T(n)[$ множество A , которое не распознается за время $T_1(n)$, если

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{T(n)} = 0.$$

Пусть M является машиной Тьюринга, которая выполняет два различных вычислительных процесса (на разных участках ленты).

(а) *Первое вычисление*: M пытается интерпретировать некоторую начальную часть входной ленты ww_b как описание машины Тьюринга M_i и затем переходит к моделированию того, что сделала бы эта машина с входной лентой ww_b . Если M завершает вычисление, а M_i допускает w , то M отвергает ее, и наоборот. Если w не описывает машины Тьюринга и вычисление не может проходить дальше, то вычисление останавливается и w отвергается. Можно показать, что для каждой ленты w (чей префикс описывает машину M_i) существует константа k_i , такая, что каждая операция M_i может быть смоделирована за k_i операций M .

(б) *Второе вычисление*. В этом вычислении M считает число операций, которые она совершает для первого вычисления, и останавливается и отвергает вход, если первое вычисление превышает $T(n)$ операций для входа w длины n . Так как $T(n)$ определяется машиной Тьюринга, M может сосчитать $T(n)$ за $T(n)$ операций. (Ради простоты мы предположим, что $T(n)$ определяется машиной, которая останавливается на каждом входе длины n за $T(n)$ операций.)

Эти два вычисления независимы, и для их одновременного проведения операции чередуются: в первом вычислении (моделировании) M отмечает ячейку ленты, на которой находится головка, «запоминает» состояние этого вычисления и тогда производит одну операцию второго вычисления (счета); после совершения этой операции M опять отмечает ячейку ленты, на которой стоит головка, и «запоминает» состояние этого вычисления; чтобы держать две «текущие позиции головки» совмещенными, M сдвигает всю запись в нижней части ленты (счет) так, чтобы совместить две головки. После этого цикл повторяется и машина колеблется между вычислениями. Поскольку $T(n)$ определяется машиной Тьюринга и вычислима на $L(n) =]\log T(n)[$ -ленте, мы видим, что на один цикл она может затратить $3]\log T(n)[$ или меньше операций. Таким образом, $T(n)$ циклов (или $T(n)$ операций) в вычислении M_i могут быть осуществлены машиной M с использованием не более чем $3T(n)]\log T(n)[$ операций. Это приводит (теорема 5) к тому, что множество A , определяемое M , распознается за время $T(n)]\log T(n)[$.

Теперь мы покажем, что если машина Тьюринга M_j работает за время $T_1(n)$ и

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{T(n)} = 0,$$

то M_j не может определять множество A , определяемое ранее описанной машиной M . Для этого заметим, что для некоторого k_j машина M моделирует одну операцию M_j за k_j операций, когда описание M_j является префиксом входа w_j , записанного на ленте машины M . В силу предельного условия существует такое N , что $k_j T_1(n) < T(n)$ для $n > N$, и поэтому для достаточно длинного входа w (префикс которого описывает M_j) у машины M достаточно времени для имитации того, что M_j должна была бы сделать со входом w , и после чего она выдает противоположный выход. Таким образом, множество, определяемое M_j , отличается от множества, определяемого M . Это завершает доказательство, так как мы установили, что за время $T(n)] \log T(n)[$ можно распознавать множество, не распознаваемое за время $T_1(n)$.

Из приведенного доказательства ясно, что множитель $] \log T(n)[$ входит в результат предыдущей теоремы ввиду необходимости совершать два независимых вычисления: имитацию M_i и счет.

Имитация использовалась для получения множества, отличающегося от всех множеств последовательностей в S_{T_1} , а операции счета были необходимы для завершения этих имитаций, и таковых требовалось больше чем $T(n)] \log T(n)[$. Кажется весьма правдоподобным, что при более глубоком проникновении в природу одноленточных вычислений мы были бы в состоянии исключить большие колебания между двумя независимыми вычислениями и уменьшить время, пропадающее в движении назад и вперед между двумя процессами.

В связи с этим интересно вспомнить, что соответствующий результат для многоленточных машин был сначала выведен в [2] и что этот результат содержал «квадрат». Только после моделирования многоленточных машин на дуоленточных машинах в [6] удалось улучшить этот результат, но даже в этом случае нам кажется, что достигнут наилучший возможный результат.

РАСПОЗНАВАНИЕ БЕСКОНТЕКСТНЫХ ЯЗЫКОВ

В этом разделе мы изучим скорость распознавания бесконтекстных языков на одноленточных машинах Тьюринга.

Лемма 4. *Существуют нерегулярные бесконтекстные языки, которые распознаются за время $T(n) = n] \log n[$ и с $R(n) =] \log n[$.*

Доказательство. Бесконтекстный язык $A = \{1^k 0^h \mid k = 0, 1, 2, \dots\}$ распознается с $R(n) = \lfloor \log n \rfloor$ и $T(n) = n \lfloor \log n \rfloor$. Напомним, что можно вычислить двоичное разложение отрезка из единиц и отрезка из нулей и проверить их совпадение за не более чем $\lfloor \log n \rfloor$ колебаний машины Тьюринга (используя лемму 2).

Таким образом, нерегулярное множество A является $(R(n) = \lfloor \log n \rfloor)$ - и $(T(n) = n \lfloor \log n \rfloor)$ -распознаваемым.

Для $W = x_1 x_2 \dots x_k$ положим $W^T = x_k \dots x_2 x_1$.

Лемма 5. Бесконтекстный язык $A = \{W \otimes W^T \mid W \in (0 \vee 1)^*\}$ является $(T(n) = n^2)$ -распознаваемым и не распознается за время $T_1(n)$, такое, что

$$\inf_{n \rightarrow \infty} \frac{T_1(n)}{n^2} = 0.$$

Доказательство. Существует простое доказательство, основанное на подсчете следов (см. [1])¹⁾. Используя функции колебаний и конструкции, аналогичные тем, которые применялись в [7, доказательство теоремы 3], можно показать, что существует бесконечно много различных вычислительных классов сложности бесконтекстных языков между временными функциями $T(n) = n \lfloor \log n \rfloor$ и $T(n) = n^2$.

Наиболее интересная проблема, остающаяся еще открытой, состоит в определении наилучшей нижней границы времени, необходимого для распознавания любого бесконтекстного языка на одноленточной машине Тьюринга. Нам известно, что эта граница должна быть не меньше чем n^2 .

Предположение автора состоит в том, что существуют бесконтекстные языки, которые нельзя распознать на одноленточной машине за время $T(n) = n^2$.

Следующий результат устанавливает верхнюю границу для распознавания бесконтекстных языков. Неизвестно, хорошая ли эта граница, но автор считает, что ее можно значительно улучшить.

Следствие 4 (Янгер). Каждый бесконтекстный язык можно распознать за время $T(n) = n^5$.

Доказательство. В [8] показано, что каждый бесконтекстный язык является $(T(n) = n^3)$ -распознаваемым на многоленточной машине Тьюринга. Прямое осуществление этого алго-

¹⁾ Этот и даже более сильный результат был одновременно и независимо получен Я. М. Барздиным [«Проблемы кибернетики», вып. 15 (1965), 245—248: «Сложность распознавания симметрии на машинах Тьюринга»] — *Прим. перев.*

ритма на одноленточных машинах Тьюринга показывает, что каждый бесконтекстный язык является $(T(n) = n^5)$ -распознаваемым.

Мы покажем, что проблема: сколько времени требуется для распознавания нерегулярных бесконтекстных языков, — является рекурсивно неразрешимой.

Теорема 8. *Не существует алгоритма для решения вопроса, может ли нерегулярный бесконтекстный язык, порожденный грамматикой G , распознаваться за время $T(n) = n \lceil \log n \rceil$.*

Доказательство. Пусть A и B суть k -наборы ненулевых двоичных слов:

$$A = (w_1, w_2, \dots, w_k),$$

$$B = (v_1, v_2, \dots, v_k), \quad v_i, w_i \in (0 \vee 1)^* - \Lambda.$$

Пусть A' и B' — такие же k -наборы в штрихованном алфавите с индексами от $k+1$ до $2k$:

$$A' = (w'_{k+1}, w'_{k+2}, \dots, w'_{2k}),$$

$$B' = (v'_{k+1}, v'_{k+2}, \dots, v'_{2k}),$$

$$v'_i, w'_i \in (0' \vee 1')^* - \Lambda \quad \text{для всех } i;$$

если $w_i = a_1 a_2 \dots a_r$, $a_j \in \{0 \vee 1\}$, то $w'_{k+i} = a'_1 a'_2 \dots a'_r$. Пусть i является двоичным разложением i -го натурального числа. Рассмотрим детерминированные бесконтекстные языки

$$L(A) = \{i_1 * i_2 * i_3 \dots * i_p * a_{i_p} \dots a_{i_2} a_{i_1} \mid p = 1, 2, \dots\},$$

$$L(B) = \{i_1 * i_2 * \dots * i_p * b_{i_p} \dots b_{i_2} b_{i_1} \mid p = 1, 2, \dots\}.$$

Тогда $L(A) \cap L(B) \neq \emptyset$ в том и только в том случае, если существует последовательность индексов $i_1, i_2, \dots, i_k$, такая, что

$$a_{i_1}^T a_{i_2}^T \dots a_{i_k}^T = b_{i_1}^T b_{i_2}^T \dots b_{i_k}^T.$$

Но проблема существования такой последовательности индексов для пары двоичных k -наборов A, B рекурсивно неразрешима [9, 10]. Таким образом, проблема пустоты $L(A) \cap L(B)$ рекурсивно неразрешима.

Напомним, что $L(A)$ и $L(B)$ — детерминированные бесконтекстные языки, и поэтому $\overline{L(A)}$ и $\overline{L(B)}$ суть бесконтекстные языки.

Таким образом,

$$\overline{L(A)} \cup \overline{L(B)} = \overline{L(A) \cap L(B)}$$

является бесконтекстным языком. Рассмотрим теперь бесконтекстный язык

$$L = \{a^n b^n \mid n = 1, 2, \dots, \} (\overline{L(AA')} \cup \overline{L(BB')}),$$

где AA' есть $2k$ -набор

$$(\omega_1, \omega_2, \dots, \omega_k, \omega'_{k+1}, \omega'_{k+2}, \dots, \omega'_{2k}).$$

Тогда из $L(AA') \cap L(BB') = \emptyset$ следует, что $\overline{L(AA')} \cup \overline{L(BB')}$ есть множество всех последовательностей над $0, 1, 0', 1', *$ и L распознается (лемма 4) за время

$$T(n) = n \lceil \log n \rceil.$$

Если $L(AA') \cap L(BB') \neq \emptyset$, то существует последовательность индексов $i_1, i_2, \dots, i_r$, такая, что

$$\begin{aligned} a_{i_1}^T a_{i_2}^T \dots a_{i_r}^T &= b_{i_1}^T b_{i_2}^T \dots b_{i_r}^T, \\ a_{i_1+k}^T a_{i_2+k}^T \dots a_{i_r+k}^T &= b_{i_1+k}^T b_{i_2+k}^T \dots b_{i_r+k}^T. \end{aligned}$$

Если мы обозначим $*i_1 * i_2 * \dots * i_r$ через E_1 , $*i_1 + k * i_2 + k * \dots * i_r + k$ через E_2 , $a_{i_1} \dots a_{i_2} a_{i_1}$ через A_1 и $a_{i_r+k} \dots a_{i_2+k} a_{i_1+k}$ через A_2 , то множество L содержит последовательности вида $E_{i_1} E_{i_2} \dots E_{i_m} * * A_{i_m} \dots A_{i_2} A_{i_1}$, где $i_j \in \{1, 2\}$, и не существует последовательности в

$$(E_1 \vee E_2)^* * * (A_1 \vee A_2)^*,$$

не имеющей такого вида. Но тогда, используя лемму 5, мы заключаем, что L требует $T(n) = n^2$ для своего распознавания. Таким образом, L распознается за время $T(n) = n \lceil \log n \rceil$ в том и только в том случае, когда $L(AA') \cap L(BB') = \emptyset$, и поэтому мы не в состоянии решить, можно ли распознать L за время $T(n) = n \lceil \log n \rceil$.

ЛИТЕРАТУРА

- 1 Hennie F. C., One-tape, off-line Turing machine computations, *Inf. Contr.*, 8 (1965), 553—578. (См. стр. 223—248 настоящего сборника.)
- 2 Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, 117 (May 1965), 285—306. (Русский перевод: Хартманис Дж., Стирнз Р., О вычислительной сложности алгоритмов, Кибернетический сборник (новая серия), вып. 4, «Мир», М., 1967.)
- 3 Stearns R. E., Hartmanis J., Lewis P. M., Hierarchies of memory limited computations, *IEEE Conf. Rec. on Switching Circuit Theory and Logical Design*, 1965, 179—190. (См. стр. 301—319 настоящего сборника.)
- 4 Трахтенброт Б. А., Тьюринговы вычисления с логарифмическим замедлением, *Алгебра и Логика, Семинар*, 3 (1964), 33—48.

5. Rabin M. O., Scott D., Finite automata and their decision problems, in Moore E. F. (Ed.) *Sequential Machines: Selected Papers*, Addison-Wesley, Mass., 1964, 63—91. (Русский перевод: Р а б и н М. О., Скотт Д., Конечные автоматы и задачи их разрешения, Кибернетический сборник, вып. 4, ИЛ, М., 1962, стр. 58—92, перевод сделан с работы, опубликованной в *IBM, J. Res. Dev.*, 3, № 2 (1959), 114—125.)
6. Hennie F. C., Stearns R. E., Two-tape simulation of multitape Turing machines, *J. ACM*, 13, № 4 (1966), 533—546 (См. стр. 194—212 настоящего сборника.)
7. Lewis P. M., Stearns R. E., Hartmanis J., Memory bounds for recognition of context-free and context-sensitive languages, *IEEE Conf. Rec. on Switching Circuit Theory and Logical Design*, 1965, 191—202. (См. стр. 320—338 настоящего сборника.)
8. Younger D. H., Context-free language processing in time n^3 , *Proc. 1966 7-th Annual Symp. on Switching and Automata Theory*, IEEE, N. Y., 1966, 7—20. (См. также статью Я н г е р а в настоящем сборнике.)
9. Post E., A variant of a recursively unsolvable problem, *Bull. Am. Math. Soc.*, 52 (1946), 262—268.
10. Ginsburg S., *The mathematical theory of context-free languages*, McGraw Hill, New York, 1966. (Русский перевод: Г и н з б у р г С., Математическая теория контекстно-свободных языков, «Мир», М., 1970.)

ИЕРАРХИИ ВЫЧИСЛЕНИЙ С ОГРАНИЧЕННОЙ ПАМЯТЬЮ ¹⁾

Р. Е. Стернз, Дж. Хартманис, П. М. Льюис II

Одна из главных целей теории автоматов состоит в том, чтобы достигнуть понимания вычислительного процесса для различных классов задач. Но теория должна идти дальше понятия вычислимости, она должна изучать, насколько сложно вычисление и как его сложность связана с устройством машины, производящей это вычисление. Две меры сложности особенно важны — это время и память. В этой работе мы получаем оценки сложности для ряда моделей машин вычислительного типа (включающего и классическую машину Тьюринга), связанные с ограничениями на ленты. Эти результаты подобны результатам (хотя и сильнее последних) об ограничениях на время, полученным в [1—3].

Использование объема ленты как меры сложности кажется менее полезным, чем оценка времени, но это компенсируется более точными результатами и многими приложениями, которые можно получить (см., например, [4]). В общем легче установить, почему задача обязательно требует так много квадратов ленты, чем почему она требует так много времени. В данной работе мы ограничимся изучением так называемых проблем разрешения, но, совершенно очевидно, эти методы можно распространить и на другие задачи.

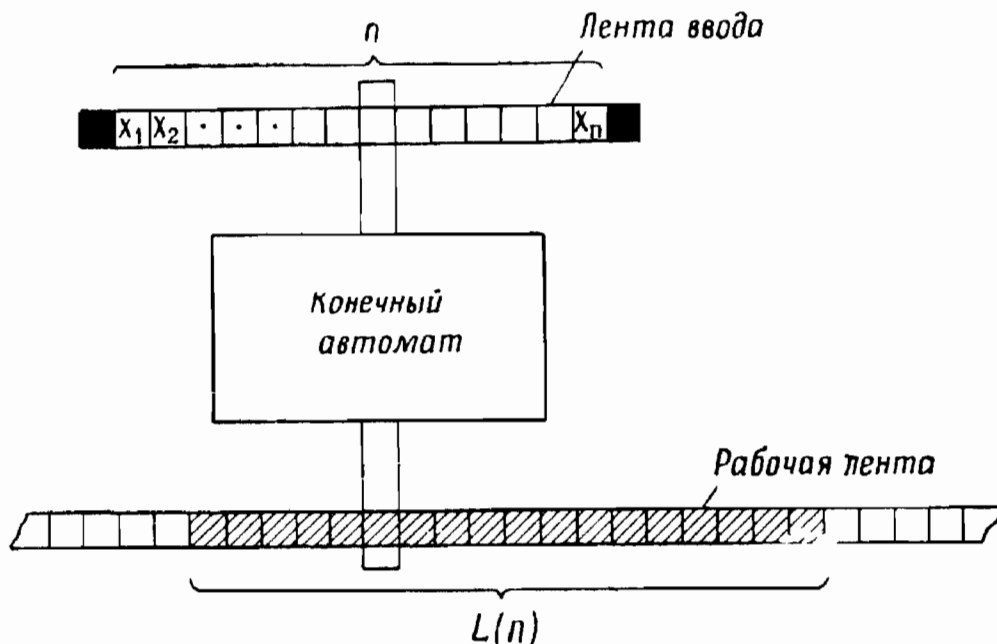
Мы рассматриваем следующие варианты машины Тьюринга: односторонние или двусторонние (on-line или off-line) и с магазинной (push-down) лентой и с обычной лентой. Лента ввода отделяется от рабочей ленты с тем, чтобы получить сложность, меньшую, чем у линейно ограниченного автомата. Исследование довольно специального случая автомата с магазинной памятью вызвано его плодотворными приложениями к изучению языков, такими, как в [4].

¹⁾ Stearns R. E., Hartmanis J., Lewis II P. M., Hierarchies of memory limited computations, IEEE Conf. Rec. Switch. Circuit Theory and Logic. Design, N. Y., 1965, 179—190.

ОСНОВНЫЕ МОДЕЛИ МАШИН

Начнем с самого общего определения — с двусторонней машины Тьюринга. Это определение особенно ориентирует нас на проблемы разрешения.

Двусторонняя машина Тьюринга (или машина с записью на ленте) — это машина, состоящая, как показано на рис. 1, из конечного управляющего устройства и двух лент: одной — только прочитываемой **ленты ввода** и другой — **рабочей ленты**, на



Р и с. 1. Двусторонняя машина Тьюринга.

которой и читают и пишут. Квадраты ленты ввода могут содержать какой-либо символ из конечного **алфавита ввода** A или специальный концевой маркерный символ; квадраты рабочей ленты могут содержать любой символ из алфавита K . Очередная операция машины определяется состоянием управляющего устройства и символами под каждой читающей головкой. Операция состоит в том, что машина печатает новый символ на рабочей ленте, сдвигает ленту ввода (на один квадрат вправо, один квадрат влево или оставляет на месте), сдвигает рабочую ленту и изменяет состояние управляющего устройства. Некоторые заключительные состояния называются **принимающими состояниями**, некоторые другие заключительные состояния называются **отвергающими состояниями**. Машина всегда начинает работу в определенном начальном состоянии с пустой рабочей лентой и читающей головкой на самом левом квадрате конечного куска ленты ввода с концевыми маркерами на обоих концевых квадратах и символами из A в других квадратах. Эти символы из A , прочитываемые слева направо, образуют **вводное слово**. Пред-

полагается, что управляющее устройство удерживает читающую головку на этом куске ленты ввода. Если последовательность операций, получаемых при некотором вводном слове, приводит машину в принимающее состояние, то будем говорить, что это слово **допускается** машиной, а если вводное слово ведет в отвергающее состояние, то будем говорить, что этот ввод **отвергается**. Если какое-то слово и не принимается и не отвергается, то будем говорить, что машина не является разрешающим устройством.

Двусторонняя магазинная машина — это двусторонняя машина Тьюринга с тем ограничением, что машина печатает пустой символ перед каждым правым продвижением своей рабочей ленты. Принято рассматривать это продвижение вправо как выборку информации, а продвижение влево — как запись.

Односторонняя машина Тьюринга (или машина со входом) — это двусторонняя машина Тьюринга с тем дополнительным ограничением, что лента ввода не может двигаться вправо, и с тем изменением, что разделенными на принимающие и отвергающие состояния будут состояния, наблюдаемые перед левым продвижением, а не заключительные состояния. Вводное слово на ленте ввода **допускается** машиной тогда и только тогда, когда машина приходит в принимающее состояние после просмотра последнего вводного символа, и машина отвергает ввод тогда и только тогда, когда она приходит в отвергающее состояние после просмотра последнего символа. Для этой односторонней модели концевые маркеры излишни.

Односторонняя магазинная машина — это односторонняя машина Тьюринга с тем ограничением, что машина печатает пустой символ перед каждым правым продвижением своей рабочей ленты.

Будем говорить, что машина M одного из вышеуказанных типов **разрешает** (распознает) множество конечных слов W тогда и только тогда, когда каждое слово из W допускается, а каждое слово не из W отвергается машиной.

ЛЕНТОЧНАЯ СЛОЖНОСТЬ

Пусть $L(n)$ — вычислимое отображение положительных целых чисел в положительные целые числа. Будем говорить, что $L(n)$ — ленточная функция и что машина M действует с лентой $L(n)$, тогда и только тогда, когда головка рабочей ленты посещает самое большее $L(n)$ квадратов рабочей ленты при любом входе длины n . Если машина M разрешает множество W и

действует с лентой $L(n)$, то будем говорить, что W $L(n)$ -ленточно разрешимо на машинах того же типа, что и M . Для каждого типа машин мы полагаем, что

$$C_{L(n)} \text{ или } C_L$$

представляют совокупность всех множеств, $L(n)$ -ленточно разрешимых на машинах этого типа. Множество $C_{L(n)}$ будет называться **классом ленточной сложности**.

Классы ленточной сложности обладают многими свойствами, общими с классами временной сложности из [1]. Для всякой $L(n)$ и всякого типа машин класс C_L рекурсивно перечислим. Поэтому ни один C_L не может содержать всех рекурсивных множеств. Для обеих моделей машины Тьюринга этот факт гарантирует бесконечное множество различных классов сложности. Главная цель этой работы состоит в более детальном исследовании этих классов сложности и их упорядочения.

Используя обозначение $[r]$ для представления наименьшего целого числа m , такого, что $m \geq r$, мы можем установить теперь основную теорему об эквивалентности.

Теорема 1. Для любого типа машин и любого положительного целого числа N множество W $L(n)$ -ленточно разрешимо тогда и только тогда, когда оно $[L(n)/N]$ -ленточно разрешимо; другими словами,

$$C_{L(n)} = C_{[L(n)/N]}.$$

Доказательство. Очевидно, что любую машину M , разрешающую W с лентой $L(n)$, можно переделать в машину M' так что один квадрат рабочей ленты M' будет содержать информацию N квадратов рабочей ленты M и что, следовательно, M' будет разрешать W с лентой $[L(n)/N]$.

Следствие 1.1. Если L и Q — ленточные функции, такие, что

$$\inf_{n \rightarrow \infty} \frac{L(n)}{Q(n)} > 0,$$

то $C_Q \subseteq C_L$.

Доказательство. Неравенство означает, что $L(n) \geq Q(n)/N$ для некоторого большого N , и потому $L(n)$ -ленточно разрешимым множеством автоматически является $[Q(n)/N]$ -ленточно разрешимое и, следовательно, $Q(n)$ -ленточно разрешимое.

Следствие 1.2. Если L и Q — ленточные функции, такие, что

$$\sup_{n \rightarrow \infty} \frac{L(n)}{Q(n)} < \infty,$$

то $C_Q \supseteq C_L$.

Доказательство. Аналогично доказательству следствия 1.1.

Следствие 1.3. Если L и Q — ленточные функции, такие, что

$$0 < \lim_{n \rightarrow \infty} \frac{L(n)}{Q(n)} < \infty,$$

то $C_Q = C_L$.

Доказательство. Искомое доказательство получается из следствий 1.1 и 1.2.

Последняя теорема и следствия показывают, что класс $C_{L(n)}$ определяется только скоростью роста $L(n)$, и потому мы можем быть менее осторожны при выборе ленточной функции. Например, можно рассматривать ленточную функцию $L(n) = \log \log n$, несмотря на то что не указано основание, что значения не целочисленны, что они просто не определены при n , равном единице. Точный способ округления этой функции до целочисленной и определения при малых n , очевидно, для нас несуществен, а изменение основания просто приводит к постоянному множителю, что по следствию 1.3 не ведет к изменению классов сложности.

МИНИМАЛЬНЫЙ РОСТ ЛЕНТОЧНОЙ ФУНКЦИИ

Наименьший класс ленточной сложности определяется ленточной функцией $L(n) = 1$ или, эквивалентно, любой постоянной или ограниченной $L(n)$. Очевидно, этот класс — класс **регулярных** множеств: эти множества можно разрешить конечным автоматом. Однако существуют и неограниченные $L(n)$, также дающие тот же класс сложности регулярных множеств. Для каждого типа машин существует минимальная скорость роста, которую нужно превысить для того, чтобы разрешить нерегулярные множества. Мы укажем эту скорость сначала для двусторонних машин.

Теорема 2. Для всякой двусторонней машины Тьюринга M , которая разрешает нерегулярное множество и действует с лентой $L(n)$, существует число p , такое, что для всякого целого числа j существует n_j , такое, что

$$L(n_j) \geq j \quad \text{и} \quad n_j \leq p^{p^j}.$$

Доказательство. Без потери общности можно предположить, что читающая головка на рабочей ленте никогда не оставляет пустым посещаемый квадрат. Это равносильно рассмотрению печатаемого пустого символа отличным от первоначальной пустоты. Определим конфигурацию памяти как комбинацию состояния управляющего устройства, строки непустых символов,

напечатанных на рабочей ленте, и положения головки на этих символах. Если строка символов длины (строго) менее j , то конфигурацию памяти назовем j -конфигурацией. Если k — число непустых символов в алфавите рабочей ленты и q — число состояний управляющего устройства, то число конфигураций памяти со строкой длины точно i есть iqk^i , а число j -конфигураций есть $\sum_{i=1}^{j-1} iqk^i$, что меньше чем jqk^j . Положив a равным числу символов в алфавите ввода, выберем p , такое, что

$$p^{p^j} > a \cdot 2^{jqk^j} \text{ для всех } j.$$

Пусть n_j — наименьшее число, такое, что M использует по крайней мере j квадратов рабочей ленты, чтобы обработать некоторое вводное слово

$$w = x_1 x_2 \dots x_{n_j}$$

длины n_j . Нерегулярность разрешаемого M множества обеспечивает существование такого слова. Из-за w получаем, конечно, что $L(n_j) \geq j$. Поэтому, когда M в конце примет или отвергнет w , она не будет находиться в j -конфигурации, так как по определению j -конфигурация использует меньше чем j квадратов рабочей ленты.

Обработывая w , машина M может много раз проходить над всяким символом x_i на ее ленте ввода. Пусть для каждого $i \leq n_j$ S_i — множество всех конфигураций, достигаемых M при обработке w , когда читающая головка находится над символом x_i .

Чтобы оценить n_j , мы покажем сначала, что из $S_r = S_s$ для $r < s$ следует, что $x_r \neq x_s$. Пусть, наоборот, $S_r = S_s$ и $x_r = x_s$ для некоторых $r < s$. Посмотрим, что произойдет при обработке машиной слова

$$w' = x_1 \dots x_r x_{s+1} \dots x_{n_j}.$$

Длина w' меньше чем n_j , и потому, согласно минимальности длины n_j , все конфигурации памяти, получаемые при переработке w' , должны быть j -конфигурациями. Покажем, что j -конфигурация для любого x_i из w' будет j -конфигурацией из множества S_i (определенного для w). Пусть, наоборот, при обработке w' M читает x_i и первый раз приходит в j -конфигурацию не из S_i . Понятно, что это может произойти, когда читающая головка переходит от x_r к x_{s+1} или от x_{s+1} к x_r . Предположим, что машина движется от x_r к x_{s+1} . По предположению ее конфигурация на x_r содержится в S_r . Но так как $S_r = S_s$, то машина будет в конфигурации, которая будет также появляться на x_s

в ω . Далее, так как $x_r = x_s$, то машина должна прийти в одну из конфигураций, появляющихся на x_s при переработке ω . Эта конфигурация по определению содержится в S_s , что доказывает предложение, что все j -конфигурации, появляющиеся при переработке ω' , из S_i . Но это означает, что M не закончит обработку ω' , так как единственная заключительная конфигурация при обработке ω не является j -конфигурацией. Так как M по определению должна останавливаться на всех вводах, то получаем противоречие, доказывающее предложение, что $S_r = S_s$ влечет $x_r \neq x_s$.

Так как S_i должны быть различны для двух вхождений одного и того же вводного символа, то n_j не должно быть больше, чем число подмножеств j -конфигураций, умноженное на число вводных символов. Но это число меньше чем $a \cdot 2^{jqk^j}$, и потому $n_j \leq p^{p^j}$ (по выбору p).

Следствие 2.1. Для двусторонней машины Тьюринга если $L(n)$ — ленточная функция, такая, что

$$\lim_{n \rightarrow \infty} \frac{L(n)}{\log \log n} = 0,$$

то C_L содержит только регулярные множества, т. е.

$$C_L(n) = C_{\text{constant}}.$$

Доказательство. Если C_L содержит нерегулярное множество, то $L(n_j) \geq j \geq \log_p \log_p n_j$ или

$$\frac{L(n_j)}{\log_p \log_p n_j} \geq 1$$

для всех n_j из теоремы и потому предел не равен нулю.

Следствие 2.2. Для машин двустороннего типа если $C_{L(n)}$ содержит нерегулярное множество, то

$$\sup_{n \rightarrow \infty} \frac{L(n)}{\log \log n} > 0.$$

Доказательство. Это есть обращение следствия 2.1.

Следствие 2.3. Для машин двустороннего типа если $L(n)$ — монотонная ленточная функция, такая, что $C_{L(n)}$ содержит нерегулярное множество, то

$$C_L(n) \supseteq C_{\log \log n}.$$

Доказательство. Положив $j = [\log_p \log_p n]$, имеем $n \geq n_j$, и потому

$$L(n) \geq L(n_j) \geq j = [\log_p \log_p n];$$

следовательно,

$$\inf_{n \rightarrow \infty} \frac{L(n)}{\log_p \log_p n} \geq 1,$$

и по следствию 1.1 получаем требуемое предложение.

Теорема 3. Для односторонней машины Тьюринга если $L(n)$ — ленточная функция, то либо

$$C_{L(n)} = C_{\text{constant}}, \quad \text{либо} \quad C_{L(n)} \equiv C_{\log n},$$

в зависимости от того,

$$\inf_{n \rightarrow \infty} \frac{L(n)}{\log(n)} = 0$$

или нет.

Доказательство. Предположим, что односторонняя машина Тьюринга M действует на ленте $L(n)$ и разрешает нерегулярное множество. Проведем все рассуждения, сделанные нами в первой части доказательства теоремы 2, за тем исключением, что p выберем таким, что

$$p^j > a \cdot j \cdot k^j$$

для всех j . Для данного n пусть

$$w = x_1 \dots x_m$$

есть наикратчайшее слово, такое, что при обработке w машиной M занимается по крайней мере $j = \log_p n$ квадратов ленты. При обработке w M не может занять j квадратов рабочей ленты, пока читающая головка не достигла x_m , иначе более короткая подпоследовательность w должна использовать j квадратов. Машина M не может повторить конфигурации на x_r и x_s , таких, что $x_r = x_s$ и $r < s$, потому что тогда

$$w' = x_1 \dots x_r x_{s+1} \dots x_m$$

было бы более коротким словом, использующим j квадратов рабочей ленты. Поэтому $m - 1$ не превосходит числа j -конфигураций, умноженного на длину алфавита ввода. Поэтому $m \leq p^j \leq n$ и w , расширенное до слова длины n , также требует $j = \log_p n$ квадратов ленты. Поэтому

$$\frac{L(n)}{\log_p n} > 1 \quad \text{для всех } n,$$

что и доказывает нужное предложение.

Теорема 4. Для односторонней магазинной машины и ленточной функции $L(n)$ либо

$$C_{L(n)} = C_{\text{constant}}, \quad \text{либо} \quad C_{L(n)} = C_n$$

в зависимости от того,

$$\inf_{n \rightarrow \infty} \frac{L(n)}{n} = 0$$

или нет.

Доказательство. Предположим, что односторонняя магазинная машина действует на ленте $L(n)$ и разрешает нерегулярное множество. Должно быть некоторое вводное слово

$$\omega = x_1 \dots x_m,$$

которое приводит машину в конфигурацию с p символами на магазинной ленте для некоторого $p > aqk$, где a — длина алфавита ввода, q — число состояний управляющего устройства и k — длина рабочего алфавита. Обозначим эти символы $k_1, \dots, k_p$. Каждому k_i соотнесем состояние управляющего устройства q_i и вводный символ x_{j_i} , который просматривается точно перед напечатанием k_i . В силу оценки p должны существовать r и s , такие, что $r < s$, $k_r = k_s$, $x_{j_r} = x_{j_s}$ и $q_r = q_s$. Если $j_r = j_s$, то машина заикнется, а если $j_r < j_s$, то

$$\omega_h = x_1 \dots x_{j_r} (x_{j_r+1} \dots x_{j_s})^h$$

есть вводное слово длины $j_r + (j_s - j_r)h$, которое требует $r + (s - r)h$ квадратов ленты. Поэтому $L(n)$ должно расти линейно, как утверждает теорема.

Подобное же рассуждение показывает, что машина не может напечатать на магазинной ленте более qk символов для данной входной буквы без заикливания и поэтому любое нерегулярное множество, которое разрешимо (на подобной машине. — *Ред.*), должно быть $(L(n) = n)$ -ленточно разрешимым.

Эта теорема полностью решает проблему для односторонней магазинной машины. Иерархия состоит из двух классов: C_1 и C_n . Эти классы различны, потому что некоторые магазинные машины разрешают нерегулярные множества.

Для односторонней машины Тьюринга очевидно, что множество

$$\{0^k 1^k \mid k \text{ — натуральное число}\}$$

$(\log n)$ -ленточно разрешимо и потому граница теоремы 3 точна.

Хотелось бы найти множества, $(\log \log n)$ -ленточно разрешимые на двусторонних машинах. Пусть b_i — двоичная запись числа i в алфавите $\{0, 1\}$, и пусть ω_k в алфавите $\{0, 1, s\}$ задается так:

$$\omega_k = b_0 s b_1 s \dots s b_k$$

(например, $\omega_4 = 0s1s10s11s100$). Чтобы узнать, содержится ли данная вводная последовательность в $\{\omega_k\}$, рассматриваем ее

как последовательность двоичных чисел, разделенных маркерами, и проверяем, совпадает ли первое число с b_0 . Если это так, начинаем проверять, верно ли, что каждое следующее число больше предыдущего на единицу. Если это не так, то обрываем вычисление и отвергаем эту последовательность. Чтобы проверить любую пару последовательных чисел с целью убедиться в том, что правое число на единицу больше левого, достаточно поразрядно сравнивать эти числа, запоминая при необходимости переносы в конечном управляющем устройстве машины. На двусторонней машине Тьюринга цифры можно сравнивать, двигаясь вперед и назад между числами и используя рабочую ленту, чтобы записывать позицию (т. е. первая, вторая и т. д.) сравниваемых цифр. Если левое число длины l , то это потребует $\log l$ рабочих квадратов. С другой стороны, легко проверить, что $l \leq \log n$, где n — длина вводного слова. Следовательно, наша схема требует $\log \log n$ на двусторонней машине Тьюринга, и нижняя граница следствия 2.3 точна для двусторонней машины Тьюринга.

Чтобы получить пример для двусторонней магазинной машины, сначала рассмотрим, как последовательность w_k можно распознать на двусторонней магазинной машине, используя $L(n) = \log n$. Цифры каждого b_i запоминаются на магазинной ленте и сравниваются затем цифра за цифрой с b_{i+1} . Теперь усовершенствуем w_k следующим образом: добавим два новых символа z и u (для нуля и единицы) к алфавиту ввода $\{0, 1\}$ и используем их для обозначения положения цифры. Итак, мы полагаем, что b_i^a — двоичная запись i в $\{0, 1\}$ с цифрами, разряды которых последовательно пронумерованы числами, записанными в $\{z, u\}$. Например,

$$b_9^a = uzzluiu0uz0u1.$$

Положим

$$w_k^a = b_0^a s b_1^a s \dots s b_k^a$$

и посмотрим, с лентой какой длины мы можем распознавать $\{w_k^a\}$. Применяя рассуждение о магазинных машинах из предыдущего раздела, легко видеть, что можно сначала проверить совпадения адресов (используя $\log l$ квадратов) и затем использовать эти адреса для сравнения знак за знаком, как в предыдущем случае; при этом весь процесс требует $\log \log n$ квадратов ленты.

МАКСИМАЛЬНЫЙ РОСТ ЛЕНТОЧНОЙ ФУНКЦИИ

Так как машины Тьюринга могут разрешить все рекурсивные множества, то нельзя найти для этих машин максимальное C_L . С другой стороны, из теоремы 4 видно, что C_n содержит все

множества, которые можно разрешить односторонней магазинной машиной. Для двусторонней магазинной машины получаем следующий результат.

Теорема 5. *Для двусторонних магазинных машин если $L(n)$ — ленточная функция, то*

$$C_{L(n)} \subseteq C_n.$$

Доказательство. Требуется показать, что C_n содержит множество, разрешимое произвольной двусторонней магазинной машиной M . Предположим, что в некоторый момент при обработке ввода длины n у машины M на рабочей ленте напечатаны символы $k_1, \dots, k_m$. Каждому символу k_i соотнесем состояние управляющего устройства q_i и квадрат ленты ввода, который просматривался сразу после того, как впервые был напечатан k_i на рабочей ленте. Если $k_r = k_s$ и $q_r = q_s$ для некоторых $r \neq s$, то соответствующие квадраты ввода должны быть различны, иначе M будет возвращаться к этому квадрату опять и опять каждый раз с дополнительными символами на ленте и никогда не остановится. Итак, должно выполняться

$$m \leq q \cdot k \cdot (n + 2),$$

где q — число состояний управляющего устройства, k — длина алфавита рабочей ленты, а $n + 2$ — число квадратов ленты ввода (включая концевые маркеры). Итак, M разрешает свое множество с лентой $L(n) \leq qk(n + 2)$, и потому по следствию 1.3 это множество $(L(n) = n)$ -ленточно разрешимо.

ПРЕДЕЛЬНЫЕ ТЕОРЕМЫ

Мы видели, что для каждого типа машин существует разрыв между ограничениями ленты для регулярных и нерегулярных множеств. Теперь мы покажем, что за этим начальным скачком (но перед C_n для магазинной машины) малейшее изменение предельного поведения данной ленточной функции дает новый класс ленточной сложности. Точное изложение этих результатов требует понятия «конструируемости», которое мы введем для каждого типа машин.

Ленточная функция $L(n)$ называется **конструируемой на двусторонней машине Тьюринга** тогда и только тогда, когда существует двусторонняя машина Тьюринга M , которая всегда останавливается, действует с лентой $L(n)$ и для каждого n занимает $L(n)$ квадратов рабочей ленты при некотором вводе длины n . Опираясь на следствия, мы позволим себе считать «конструируемыми» функции $Q(n)$ с таким же предельным поведением,

как и конструируемые $L(n)$. Это означает, например, что $L(n) = \log \log n$ конструируема, потому что уже была указана машина (для разрешения $\{\omega_n\}$), которая действует с лентой $\log \log n$ и занимает «точно» $\log \log n$ для некоторых вводов длины n . Подобным образом каждая двусторонняя машина Тьюринга, разрешающая некоторое множество, определяет также соответствующую конструируемую ленточную функцию. «Вычислимые в реальное время» функции в работах [1, 5] конструируемы. Если $R(n)$ вычислима в реальное время, то можно сначала получить $\log \log n$ по вышеуказанной схеме и затем отсчитать $R(\log \log n)$ квадратов рабочей ленты. Поэтому $R(\log \log n)$ конструируема. Так как известно, что вычислимые в реальное время функции включают все общеупотребительные монотонно возрастающие функции, получаемые умножением, экспоненцированием и суперпозицией, то в действительности иерархия конструируемых функций, построенная исходя из $\log \log n$, весьма богата. Покажем теперь, что иерархии конструируемых функций соответствует иерархия классов ленточной сложности (для двусторонних машин Тьюринга) с учетом эквивалентности в следствиях 1.1 и 1.2.

Теорема 6. *Для двусторонних машин Тьюринга если $L(n)$ — конструируемая ленточная функция, то существует множество W , которое $L(n)$ -ленточно вычислимо и не является $Q(n)$ -ленточно вычислимым для всех ленточных функций $Q(n)$, таких, что*

$$\inf_{n \rightarrow \infty} \frac{Q(n)}{L(n)} = 0.$$

Доказательство. Доказываем диагональным методом. Построим машину M , которая действует на ленте $L(n)$, такую, что для любой другой машины, использующей ленту $Q(n)$ (такую, как в формулировке теоремы), существует некоторая вводная последовательность, на которой M повторяет действия этой другой машины на этой же последовательности и дает отличный выход. Множество W и есть множество, разрешаемое на M .

Пусть M' — машина, конструирующая $L(n)$ и обладающая алфавитом ввода A' . Рассмотрим новый алфавит A , получаемый из A' следующим образом: каждому символу x из A' отвечают два символа x_0 и x_1 из A . Каждое вводное слово в A можно рассматривать как слово в A' (опуская индексы) или как двоичное число (читая только индексы). Теперь построим машину M , которая работает в три этапа.

Этап 1. Машина M читает ввод в A как слово в A' и действует подобно M' , пока M' не остановится. Во время работы

она помечает самые левый и правый из квадратов рабочей ленты, которые посетила. При любых обстоятельствах M будет ограничиваться квадратами между метками, гарантируя тем самым, что действует с лентой $L(n)$.

Этап 2. Читая ввод как двоичное число i , M печатает на своей рабочей ленте описание i -й двусторонней машины Тьюринга в алфавите ввода A . Это потребует определенное количество D_i квадратов. Если окажется, что места между концевыми маркерами на рабочей ленте недостаточно, то машина останавливается и приходит в отвергающее состояние.

Этап 3. Если описание M_i успешно завершилось, то машина M начинает моделировать M_i , используя ленту ввода как ввод и специальную дорожку рабочей ленты как рабочую ленту M_i . Символы рабочей ленты M_i кодируются в двоичном алфавите и в общем потребуется некоторое k_i квадратов M , чтобы разместить информацию одного квадрата M_i . Третья дорожка рабочей ленты используется для подсчета в троичной системе количества промоделированных операций M_i . Моделирование продолжается, пока не наступит одна из следующих трех ситуаций.

а) Моделирование требует больше рабочей ленты, чем допускается концевыми маркерами. В этом случае M приходит в отвергающее состояние.

б) Подсчет операций M_i требует больше рабочей ленты, чем допускается концевыми маркерами. В этом случае M приходит в отвергающее состояние. Заметим, что это правило гарантирует, что M всегда остановится.

в) Машина M_i останавливается. Машина M приходит в отвергающее состояние, если M_i пришла в принимающее состояние, и M приходит в принимающее состояние, если M_i пришла в отвергающее состояние.

Очевидно, машина M разрешает некоторое множество W с лентой $L(n)$, и остается только показать, что W не может быть разрешимым с любой функцией $Q(n)$, которая удовлетворяет теореме. Предположим, наоборот, что некоторая M_j разрешает W с лентой $Q(n)$, где $Q(n)$ — некоторая функция, удовлетворяющая предельному равенству. Ясно, что существует столь большое число N , что

$$L(N) \geq \log_2 j, \quad D_j \leq L(N), \quad k_j Q(N) \leq L(N),$$

$$g_j \cdot L(N) \cdot 2^{L(N)} < 3^{L(N)},$$

где g_j — число состояний управляющего устройства M_j . Пусть w' — слово в A' , имеющее длину N , для которого M' посещает

$L(N)$ квадратов рабочей ленты. Добавив индексы, преобразуем w' в слово w в A , такое, что представляемое двоичное число есть j . Это возможно потому, что $L(N) \geq \log_2 j$, и потому, что можно прибавить достаточно нулей слева к двоичному представлению j , чтобы получить двоичное представление длины N . Посмотрим, что произойдет, если ввод w обработать M .

На этапе 1 будут отмечены $L(N)$ квадратов ленты. Этап 2 сможет завершиться, так как $D_j \leq L(N)$. На этапе 3 ситуация а) невозможна, так как M_j потребует только $k_j \cdot Q(N)$ квадратов ленты, что меньше $L(N)$. Ситуация б) невозможна, так как M_j не закикливается и поэтому не может сделать более чем $L(N) g_j 2^{L(N)}$ операций перед остановкой, а M может считать до $3^{L(N)}$ операций. Как только наступает ситуация в), M отвергает те слова, что принимает M_j , и принимает те, что отвергает M_j , а это будет противоречить допущению, что M_j разрешает w . Итак, неверно, что M_j разрешает W с лентой $Q(n)$, что и доказывает теорему.

Ленточная функция $L(n)$ называется конструируемой на односторонней машине Тьюринга тогда и только тогда, когда существует односторонняя машина Тьюринга, которая всегда останавливается, действует с лентой $L(n)$ и для каждого n занимает точно $L(n)$ квадратов рабочей ленты для всех вводов длины n . Опять мы, допуская нестрогость, назовем $Q(n)$ конструируемой, если она имеет то же предельное поведение, что и $L(n)$. Это означает, например, что $L(n) = \log n$ конструируема, так как подсчет числа входных символов при соответствующем основании b заставляет машину посещать точно $\lceil \log_b n \rceil$ квадратов. Подобно двустороннему случаю, если $R(n)$ — вычислимая в реальное время функция, то функция $R(\log n)$ конструируема. Итак, имеется богатая иерархия конструируемых функций, построенная исходя из $\log n$, которая связана, как мы покажем, с иерархией ленточной сложности для односторонних машин Тьюринга.

Теорема 7. Для односторонних машин Тьюринга если $L(n)$ — конструируемая ленточная функция, то существует множество W , которое $L(n)$ -ленточно вычислимо и не является $Q(n)$ -ленточно вычислимым для всех ленточных функций $Q(n)$, таких, что

$$\inf_{n \rightarrow \infty} \frac{Q(n)}{L(n)} = 0.$$

Доказательство. Так как теорема тривиально выполняется для ограниченной $L(n)$, можно полагать, что $L(n)$ неограниченная¹⁾. Мы знаем из доказательства теоремы 4, что для

¹⁾ Аналогичное замечание необходимо при доказательстве теоремы 6. — Прим. перев.

любой машины, конструирующей неограниченную $L(n)$, должно выполняться $L(n) \geq \log n$ при некотором основании логарифмов. Это дает нам возможность снова использовать диагонализационную процедуру. Мы построим одностороннюю машину M , использующую ленту $L(n)$, такую, что для произвольной машины, которая использует ленту $Q(n)$, существует некоторая вводная последовательность, на которой M моделирует эту машину, но дает отличный выход. Из-за одностороннего характера M моделирование проводится несколько необычно.

Алфавит ввода $M = \{0, 1\}$. После своего i -го вводного символа M должна сделать три манипуляции.

Этап 1. Машина M действует как машина, конструирующая $L(n)$, и отмечает $L(i)$ квадратов на рабочей ленте, используя специальные концевые маркеры. Последующие машинные операции перед следующим вводным символом будут ограничиваться этим пространством.

Этап 2. Машина заполняет свою память ввода. Просмотренный ввод записывается (если это возможно) на рабочей ленте как некоторое число и как некоторое слово на соответствующих двух дорожках. Число выражает количество начинающих вводную последовательность нулей, слово состоит из первой единицы и всех последующих вводных символов. Количество начальных нулей можно всегда запоминать, кодируя его при некотором основании b , таком, что $L(n) \geq \log_b n$. Если слово не уместится на предназначенном пространстве, то машина приходит в состояние, отвергающее настоящий и все последующие вводные символы.

Этап 3. Трактую записанное слово как двоичное число j , машина рассматривает его как описание j -й односторонней машины Тьюринга M_j с двсичным вводом и моделирует M_j , как если бы та обрабатывала вводную последовательность, уже предложенную M . При использовании специальной двоичной дорожки рабочей ленты для моделирования рабочей ленты M_j будет достаточно k_j квадратов M , чтобы записать информацию одного квадрата M_j . Число операций M_j для каждого вводного символа подсчитывается в троичной системе на другой дорожке рабочей ленты. Моделирование продолжается, пока не наступит одна из следующих трех ситуаций.

а) Моделирование требует больше рабочей ленты, чем допускается концевыми маркерами. В этом случае M приходит в отвергающее состояние и сдвигается на следующий вводный символ.

б) Подсчет операций M_j требует больше рабочей ленты, чем допускается концевыми маркерами. В этом случае M приходит в отвергающее состояние и сдвигается на следующий вводный

символ. Это правило предохраняет M от бесконечной обработки вводного символа.

в) Машина M_j завершает обработку вводного слова. Машина M отвергает ввод, если M_j принимает его, и принимает ввод, если M_j отвергает его. Затем машина M сдвигается на следующий вводный символ.

Очевидно, что машина M разрешает некоторое множество W с лентой $L(n)$; остается только показать, что W удовлетворяет теореме. Предположив, наоборот, что некоторое M_j разрешает W с лентой $Q(n)$, где $Q(n)$ — некоторая функция, удовлетворяющая предельному равенству, найдем N , такое, что¹⁾

$$L(N - [\log_2 j]) \geq \log_2 j, \quad k_j Q(N) \leq L(N), \quad g_i \cdot L(N) \cdot 2^{L(N)} \leq 3^{L(N)}.$$

Посмотрим, что делает M после получения $N - [\log_2 j]$ нулей, за которыми следует двоичное представление j . Первое неравенство гарантирует наличие места, достаточного для заполнения ввода, второе гарантирует наличие места, достаточного для моделирования M_j , а третье гарантирует, что операции M_j не переполнят вычислитель.

Следуя рассуждению, использованному в доказательстве теоремы 6, получаем, что в действительности M и M_j трактуют этот ввод противоположным образом. Полученное противоречие и доказывает теорему.

Для двусторонних магазинных машин нам неизвестна возможность использовать диагонализационное доказательство, как было сделано для машин Тьюринга, но мы все-таки можем указать иерархию построением множеств, удовлетворяющих следующей лемме.

Лемма. Если W — множество слов в алфавите A и $L(n)$ — отображение целых чисел в целые числа, причем для любого n существуют слово w_n из W и символ a из A , такие, что

1) последние $L(n)$ символов в w_n суть a ,

2) w_n с присоединенной справа любой последовательностью символов a не содержится в W ,

и если W $Q(n)$ -ленточно вычислимо на двусторонней магазинной машине M , то существует такая постоянная c , что $Q(n) \geq cL(n)$.

Доказательство. У нас слишком мало места, чтобы привести все детали доказательства, поэтому изложим основную идею. Пусть q — число состояний управляющего устройства M , k — число символов рабочего алфавита. Можно показать, что,

¹⁾ Здесь k_j и g_j трактуются как в доказательстве теоремы 6. — Прим. перев.

когда M обрабатывает w_n и обозревает часть ввода, содержащую $L(n)$ символов a , машина не может сделать более $q \cdot k^{qk} Q(n)$ операций до того, как ее поведение станет «периодическим» с периодом, не превосходящим $q \cdot k^{qk}$. Эта периодичность заключается либо в периодическом повторении конфигурации памяти (состояние и рабочая лента), когда длина ленты одна и та же после каждого периода, либо в периодическом возрастании, когда число символов на ленте увеличивается после каждого периода. Если машина уже стала периодически повторяться, когда она проходит над символами a , то она не различит w_n и w_n с присоединенными $(qk^{qk})!$ символами a , потому что эти лишние a кратны периоду. Единственная альтернатива состоит в том, что либо M не становится периодической и в этом случае

$$qk^{qk} Qn \geq L(n),$$

либо M становится периодически возрастающей. Но если M становится периодически возрастающей, то можно оценить число операций, так как длина ленты не может превзойти $Q(n)$. После самое большее $qk^{qk} Q(n)$ операций должно начаться периодическое возрастание, и с этого момента после каждого самого большего qk^{qk} операций длина ленты должна возрасти по крайней мере на 1. Так как может быть самое большее $Q(n)$ скачков, то число операций самое большее

$$qk^{qk} Q(n) + qk^{qk} Q(n) = 2qk^{qk} Q(n).$$

Но это число операций должно быть достаточным, чтобы ободреть $L(n)$ символов a хотя бы раз, и потому должно выполняться

$$2qk^{qk} Q(n) \geq L(n).$$

Итак, в обоих случаях лемма справедлива, если выбрать $1/c = 2qk^{qk}$.

Будем говорить, что ленточная функция $L(n)$ конструируема на двусторонней магазинной машине, если существует множество W , которое удовлетворяет лемме и $L(n)$ -ленточно разрешимо на двусторонней магазинной машине. Менее формально, $L(n)$ конструируема, если для некоторого ввода длины n можно узнать, имеются ли на его конце в точности $L(n)$ единиц, и это узнавание можно произвести с лентой $L(n)$. Хотя эти конструируемые функции детально не исследованы, известно, что таковыми являются различные рациональные степени n , $\log n$ и $\log \log n$, лежащие между n и $\log \log n$, что дает довольно богатую структуру. Перенесем эту структуру на иерархию сложности.

Теорема 8. Для двусторонних магазинных машин если $L(n)$ — конструируемая функция, то существует множество W , которое

$L(n)$ -ленточно вычислимо и не является $Q(n)$ -ленточно вычислимым для всех ленточных функций $Q(n)$, таких, что

$$\inf_{n \rightarrow \infty} \frac{Q(n)}{L(n)} = 0.$$

Доказательство. В качестве W возьмем множество, делающее $L(n)$ конструируемой. Если W также $Q(n)$ -ленточно вычислимо, то лемма гарантирует нам, что $Q(n) \geq cL(n)$ для некоторого c и потому

$$\inf_{n \rightarrow \infty} \frac{Q(n)}{L(n)} \geq c.$$

ВЫВОДЫ

Мы видели, что ограничение ленты, требуемой для разрешения множества на любой из машин четырех типов, может служить мерой сложности проблемы разрешения. Для трех моделей это ведет к богатой иерархии классов ленточной сложности, представляющих различную степень трудности. Структура этих иерархий такова:

1) Для каждой модели «простейший» класс сложности — это C_{constant} , являющийся классом всех регулярных множеств.

2) Каждый тип машины обладает нижней границей $\mathcal{L}(n)$, которая ограничивает нерегулярные классы сложности в том смысле, что

а) $\lim_{n \rightarrow \infty} \frac{L(n)}{\mathcal{L}(n)} = 0$ влечет, что C_L содержит только регулярные множества;

б) $C_{L(n)} \equiv C_{\mathcal{L}(n)}$ для всех монотонных $L(n)$, таких, что C_L содержит нерегулярные множества.

При этом функция $\mathcal{L}(n)$ такова:

двусторонние машины Тьюринга	$\log \log n$,
двусторонние магазинные машины	$\log \log n$,
односторонние машины Тьюринга	$\log n$,
односторонние магазинные машины	n .

3) В случае машин Тьюринга существует бесконечная иерархия классов, построенная исходя из $\mathcal{L}(n)$ и такая, что малейшее изменение (в сторону возрастания) предельного поведения одной ленточной функции дает другую ленточную функцию, определяющую новый класс сложности (когда новая функция конструируема).

4) Для двусторонних магазинных машин существует максимальный класс сложности, а именно $C_{L(n)=n}$, содержащий все

множества, которые можно разрешить на двусторонней магазинной машине. Между функциями $L(n)=n$ и $\mathcal{L}(n)=\log \log n$ подходящее малое изменение (в сторону возрастания) предельного поведения дает опять новый класс.

5) Для односторонней магазинной машины иерархия состоит из двух множеств: $C_{L(n)=n}$ и C_{constant} .

ЛИТЕРАТУРА

1. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. of Amer. Math. Soc.*, **117** (1965), 285—306. (Русский перевод: Кибернетический сборник, новая серия, вып. 4, 1967.)
2. Hartmanis J., Stearns R. E., Computational complexity of recursive sequences, *Proc. Fifth Annual Sympos. on Switching Theory and Logical Design*, Princeton, N. J., 1964.
3. Hennie F. C., Stearns R. E., Two-tape simulation of multi-tape Turing machines, General Electric Report 65-RL-4020 E¹).
4. Lewis II P. M., Stearns R. E., Hartmanis J., Memory bounds for the Recognition of context free and context sensitive languages, *IEEE Conf. Rec. Switch. Circuit Theory and Logical Design*, 1965, 191—202. (Настоящий сборник, стр. 320—338.)
5. Yamada H., Real-time computation and recursive functions not real-time computable, *IRE Trans.*, **EC-11** (1962), 753—760. (Настоящий сборник, стр. 139—155.)
6. Turing A. M., On computable numbers, with applications to the Entscheidungsproblem, *Proc. London Math. Soc.* (22), **42** (1937), 230—265.
7. Myhill J., Linear bounded automata, WADD Tech. Note 60-165, Rep. № 60-22, Univ. of Pennsylvania, June 1960.
8. Ritchie R. W., Classes of predictably computable functions, *Trans. Amer. Math. Soc.*, **106** (1963), 139—173. (Настоящий сборник, стр. 50—93.)
9. Hartmanis J., Lewis II P. M., Stearns R. E., Classifications of computations by time and memory requirements, *Proc. of IFIP Congress 65*, vol. 1, 31—35.

¹) См. также одноименную статью тех же авторов в *J. ACM*, **13**, № 4 (1966), 535—546, перевод которой включен в настоящий сборник (стр. 194—212). — Прим. ред.

ГРАНИЦЫ ПАМЯТИ ДЛЯ РАЗРЕШЕНИЯ КОНТЕКСТНО-СВОБОДНЫХ И КОНТЕКСТНЫХ ЯЗЫКОВ ¹⁾

П. М. Льюис II, Р. Е. Стернз, Дж. Хартманис

1. ВВЕДЕНИЕ

Важной частью теории абстрактных языков является изучение процесса обработки этих языков на машинах различных моделей. Результаты этой области не только раскрывают важные свойства языка, но и дают непосредственные приложения также в изучении языков вычислительных машин.

В настоящей работе рассматриваются проблемы разрешения контекстно-свободных и контекстных языков машинами различного устройства.

Будем говорить, что машина разрешает данный язык, если, как только цепочка символов размещается на ее ленте ввода, машина обрабатывает эту цепочку и обязательно останавливается, выдавая «да», если цепочка содержится в языке, и «нет» в противном случае.

Опыт показывает, что некоторые языки существенно сложнее для разрешения, чем другие, и что эта сложность может сильно зависеть от устройства обрабатывающей машины.

Сложность вычисления можно измерять различным образом. В этой работе мы используем как меру сложности объем памяти, необходимой для вычисления; более точно $L(n)$ -ленточная мера сложности определена в [1]. Чтобы изучить зависимость этой меры сложности от устройства машины, рассмотрим четыре модели машин, определенные в [1], а именно: двустороннюю и одностороннюю машины Тьюринга, двустороннюю и одностороннюю магазинные машины. Если какая-либо из этих машин разрешает данный язык и обрабатывает каждое вводное слово длины n , используя не более $L(n)$ квадратов своей рабочей ленты, то будем называть этот язык $L(n)$ -ленточно разрешимым или просто $L(n)$ -разрешимым. Так как мы изучаем ленточные функции с точки зрения скорости роста, мы ограничимся монотонными неубывающими ленточными функциями.

Результаты [1] позволяют естественно ввести частичное упорядочение « $\leq$ » ленточных функций. Будем писать $L(n) \leq Q(n)$

¹⁾ Lewis II P. M., Stearns R. E., Hartmanis J., Memory bounds for recognition of context-free and context-sensitive languages, IEEE Conf. Rec. Switch Circuit Theory and Logic. Design, N. Y., 1965, 191—202.

тогда и только тогда, когда существует константа C , такая, что $L(n) \leq CQ(n)$ для всех n . Будем писать $L(n) \doteq Q(n)$ тогда и только тогда, когда $L(n) \leq \cdot Q(n)$ и $Q(n) \leq \cdot L(n)$. Наконец, будем писать $L(n) < \cdot Q(n)$, если неверно, что $Q(n) \leq \cdot L(n)$. Из [1] нам известно, что $Q(n) \leq \cdot L(n)$ означает, что все $Q(n)$ -разрешимые множества $L(n)$ -разрешимы, $L(n) \doteq Q(n)$ означает, что класс $Q(n)$ -разрешимых множеств и класс $L(n)$ -разрешимых множеств одинаковы, и $L(n) < \cdot Q(n)$ означает, что некоторые $Q(n)$ -разрешимые множества не являются $L(n)$ -разрешимыми, если $Q(n)$ конструируема. Так как $\leq \cdot$ есть только частичное упорядочение, то вполне возможно, что $L(n) < \cdot Q(n)$ и $Q(n) < \cdot L(n)$.

2. ОПРЕДЕЛЕНИЕ АБСТРАКТНЫХ ЯЗЫКОВ

Рассматриваемые нами абстрактные языки были предложены Хомским [2]. Они определяются в терминах порождающих грамматик. Грамматика состоит из двух непересекающихся конечных множеств символов,

$N = \{S, A, B, C, \dots\}$ — множества нетерминальных символов,
 $T = \{a, b, c, \dots\}$ — множества терминальных символов,
 S — выделенного в N начального символа и
 P — конечного множества правил.

Мы рассматриваем три типа грамматик, определяемые видом их правил:

1. Контекстная грамматика — это грамматика с правилами вида

$$\alpha A \beta \rightarrow \alpha \gamma \beta,$$

где α, β, γ — слова из терминальных и нетерминальных символов и γ не пусто. Это правило трактуется так: в любом слове нетерминальный символ A , находящийся между α и β , можно заменять на γ .

2. Контекстно-свободная грамматика — это грамматика с правилами вида

$$A \rightarrow \gamma.$$

3. Автоматная (finite state) или регулярная грамматика — это грамматика, такая, что либо все ее правила вида

$$\begin{aligned} A &\rightarrow aB, \\ A &\rightarrow a, \end{aligned}$$

либо все ее правила вида

$$\begin{aligned} A &\rightarrow Ba, \\ A &\rightarrow a, \end{aligned}$$

т. е. один нетерминальный символ можно заменять либо на один терминальный, либо на один терминальный с присоединенным справа или слева нетерминальным символом.

Язык $L(G)$, определяемый грамматикой G , — это множество слов в T^* , которые можно породить этой грамматикой, исходя из начального символа S . (* всегда обозначает оператор навешивания звездочки по Клини.) Язык называется контекстным (или контекстно-свободным или регулярным), если существует по крайней мере одна порождающая его контекстная (или контекстно-свободная или регулярная) грамматика.

Класс контекстно-свободных языков особенно интересен, так как включает АЛГОЛ и другие языки программирования.

3. РАЗРЕШЕНИЕ АБСТРАКТНЫХ ЯЗЫКОВ

Здесь излагаются некоторые известные результаты о разрешении абстрактных языков.

А. Контекстные языки. Множество предложений является контекстным языком тогда и только тогда, когда его можно разрешить недетерминированным линейно ограниченным автоматом (Ландвебер [3] и Курода [4]).

В нашей терминологии линейно ограниченный автомат — это двусторонняя машина Тьюринга с $L(n) = n$. Отсюда следует, что множество, разрешимое какой-либо из моделей машин с $L(n) \leq \cdot n$, является контекстным языком.

Из [1] следует, что как модели односторонней и двусторонней машин Тьюринга, так и модель двусторонней магазинной машины определяют иерархию различных классов сложности контекстных языков. Для двусторонней машины Тьюринга и двусторонней магазинной машины эти иерархии покрывают следующую область для $L(n)$:

$$\log \log n \leq \cdot L(n) \leq \cdot n,$$

а для односторонней машины Тьюринга иерархия покрывает область

$$\log n \leq \cdot L(n) \leq \cdot n.$$

Напомним, например, множества

$$\mathcal{S}_1 = \{sb_1sb_2s \dots sb_ms \mid m = 1, 2, \dots\},$$

$$\mathcal{S}_2 = \{sb_1^a sb_2^a s \dots sb_m^a s \mid m = 1, 2, \dots\},$$

введенные в [1], где b_j — двоичное представление числа j , а b_j^a — такое же представление с последовательно пронумерован-

ными цифрами. Множества $\mathcal{E}_1$ и $\mathcal{E}_2$ суть такие контекстные (но не контекстно-свободные) языки, которые можно разрешить с лентой

$$L(n) = \log \log n$$

на двусторонней машине Тьюринга и двусторонней магазинной машине соответственно.

Б. Контекстно-свободные языки. Контекстно-свободные языки часто изучаются совместно с магазинными машинами из-за следующих обстоятельств.

1. Множество предложений является контекстно-свободным языком тогда и только тогда, когда его можно разрешить недетерминированной односторонней магазинной машиной. Существует контекстно-свободный язык, который нельзя разрешить никакой детерминированной односторонней магазинной машиной (Хомский [5]).

Поэтому класс языков, разрешимых односторонними магазинными машинами, образует собственное подмножество класса контекстно-свободных языков. Как показано в [1], существует единственный класс сложности, содержащий нерегулярные множества, а именно класс $(L(n) = n)$ -разрешимых множеств.

2. Двусторонние магазинные машины могут разрешать и контекстно-свободные и контекстные языки. Например, множество предложений

$$\{a^n b^n c^n \mid n = 1, 2, \dots\},$$

контекстное, но не контекстно-свободное, можно разрешить двусторонней магазинной машиной. Мы высказываем гипотезу, что совокупность множеств, разрешимых двусторонними магазинными машинами, строго включается в множество контекстных языков и не содержит всех контекстно-свободных языков.

3. Все контекстно-свободные языки можно разрешить линейно ограниченным автоматом (Курока [4]).

Поэтому все контекстно-свободные языки можно разрешить двусторонними машинами Тьюринга с $L(n) \leq n$. Результаты следующего раздела дают более точную оценку $L(n)$, необходимой для разрешения всех контекстно-свободных языков на двусторонних машинах Тьюринга.

В. Регулярные языки. Язык регулярен тогда и только тогда, когда его можно разрешить конечным автоматом (Хомский [2]). Отсюда следует, что регулярные языки можно разрешить на любой модели машин с $L(n) = 1$.

4. ИЕРАРХИИ КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ

В этом разделе мы покажем, что двусторонние и односторонние машины Тьюринга и двусторонние магазинные машины определяют бесконечные иерархии контекстно-свободных языков. Как обсуждалось выше, односторонняя магазинная машина обладает только одним классом сложности $L(n) = n$ и потому не определяет иерархию. Уже было показано, что машины Тьюринга включают все контекстно-свободные языки в свои иерархии.

Чтобы дать полную картину различных результатов об иерархиях, приведем сначала все формулировки и только затем дадим доказательства.

А. Для двусторонней машины Тьюринга.

1. Существуют $(L(n) = \log \log n)$ -разрешимые нерегулярные контекстно-свободные языки.

2. Существуют контекстно-свободные языки, требующие $L(n) \doteq \log n$.

3. Существует бесконечная иерархия контекстно-свободных языков по некоторым ленточным функциям $L(n)$, удовлетворяющим условию

$$\log \log n \leqslant \cdot L(n) \leqslant \cdot \log n.$$

4. Все контекстно-свободные языки можно разрешить с

$$L(n) = (\log n)^2.$$

Иерархия указывается только для области между $\log \log n$ и $\log n$ и не доводится до границы $(\log n)^2$, данной в 4. Остается открытым вопрос, существуют ли контекстно-свободные языки, требующие больше чем $L(n) = \log n$ на двусторонних машинах Тьюринга. Мы предполагаем, что такие языки существуют и что $(\log n)^2$ — настоящая граница.

Б. Для односторонней машины Тьюринга.

1. Существуют $(L(n) = \log n)$ -разрешимые нерегулярные контекстно-свободные языки.

2. Существуют контекстно-свободные языки, требующие $L(n) \doteq n$.

3. Существует бесконечная иерархия контекстно-свободных языков по некоторым ленточным функциям $L(n)$, удовлетворяющим условию

$$\log n \leqslant \cdot L(n) \leqslant \cdot n.$$

В. Для двусторонней магазинной машины.

1. Существуют $(L(n) = \log \log n)$ -разрешимые нерегулярные контекстно-свободные языки.

2. Существуют контекстно-свободные языки, требующие $L(n) \doteq n$.

3. Существует бесконечная иерархия контекстно-свободных языков по некоторым ленточным функциям $L(n)$, удовлетворяющим условию

$$\log \log n \leqslant \cdot L(n) \leqslant \cdot n.$$

Определим три языка, которые будут служить примерами нерегулярных контекстно-свободных языков, разрешимых с $L(n)$, указанной соответственно в предложениях **A1**, **B1**, **B1**.

Пусть L_1 — язык со следующим дополнением:

$$\bar{L}_1 = \{\vec{b}_1 s \overleftarrow{b}_2 s \vec{b}_3 s \dots s \overleftarrow{b}_{2m} \mid m = 1, 2, \dots\},$$

где $\vec{b}_j$ есть двоичное представление числа j , выписанное слева направо; $\overleftarrow{b}_k$ — двоичное представление k , выписанное справа налево; s — разделительный символ.

Пусть L_2 — язык с дополнением

$$\bar{L}_2 = \{\vec{b}_1^a s \overleftarrow{b}_2^a s \vec{b}_3^a s \dots s \overleftarrow{b}_{2m}^a \mid m = 1, 2, \dots\},$$

где $\vec{b}_j^a$ — двоичное представление j , выписанное слева направо, с последовательно пронумерованными цифрами, причем номера выписаны в противоположном направлении (см. также примеры из [1]). Например,

$$\vec{b}_{25}^a = uzulzzuluu0zu0u1,$$

где в номерах позиций цифр z служит нулем, а u единицей. Аналогично определяется $\overleftarrow{b}_k^a$.

Наконец, пусть

$$L_3 = \{a^n b^n \mid n = 1, 2, \dots\}.$$

Теорема 1. Множества L_1 , L_2 и L_3 суть нерегулярные контекстно-свободные языки.

Язык L_1 ($L(n) = \log \log n$)-разрешим на двусторонней машине Тьюринга.

Язык L_2 ($L(n) = \log \log n$)-разрешим на двусторонней магазинной машине.

Язык L_3 ($L(n) = \log n$)-разрешим на односторонней машине Тьюринга.

Доказательство. Очевидно, языки L_1 , L_2 и L_3 нерегулярны.

И L_1 и L_2 можно разрешить на соответствующих машинах с $L(n) = \log \log n$, используя очевидную модификацию процедуры обработки примеров из [1]. Остается только показать, что L_1 и L_2 — контекстно-свободные языки.

Мы дадим доказательство для L_1 ; доказательство для L_2 совершенно аналогично.

Легко проверить, что $\bar{L}_1$ можно представить как пересечение двух других языков¹⁾

$$\bar{L}_1 = L_4 \cap L_5,$$

где

$$L_4 = \vec{b}_1 s (\vec{b}_j s \vec{b}_{j+1} s)^* (0 + 1)^* s,$$

$$L_5 = (\vec{b}_k s \vec{b}_{k+1} s)^*.$$

Тогда L_1 можно представить так:

$$L_1 = \overline{(L_4 \cap L_5)} = \bar{L}_4 \cup \bar{L}_5.$$

Мы покажем, что L_4 и L_5 суть контекстно-свободные языки, разрешимые односторонней магазинной машиной. Поэтому дополнения к ним — также контекстно-свободные языки (потому что выход машины можно заменить на противоположный). Так как объединение двух контекстно-свободных языков есть контекстно-свободный язык, L_1 контекстно-свободен.

Чтобы показать, например, что L_5 можно разрешить односторонней магазинной машиной, рассмотрим следующее неформальное описание разрешающей процедуры. Просмотрев $\vec{b}_k$, машина запоминает это слово на магазинной ленте. После того как машина увидела символ s , последовательность $\vec{b}_k$ сдвигается в противоположном направлении на одну цифру за каждый шаг и обрабатывается при этом последовательным сумматором (содержащимся в конечной части машины), который прибавляет 1 к этой последовательности. Выход последовательного сумматора можно сравнивать цифра за цифрой с $\vec{b}_{k+1}$, что завершает разрешение. Процедура повторяется для каждой скобки под оператором $*$.

Подобную же процедуру можно использовать для разрешения L_4 односторонней магазинной машиной. Этим завершается доказательство для двусторонних машин.

¹⁾ Здесь $(\vec{b}_k s \vec{b}_{k+1} s)^*$ обозначает множество T^* , где $T = \{\vec{b}_k s \vec{b}_{k+1} s \mid k = 1, 2, \dots\}$; аналогично понимается $(\vec{b}_j s \vec{b}_{j+1} s)^*$. — Прим. перев.

Очевидно, язык

$$L_3 = \{a^n b^n \mid n = 1, 2, \dots\}$$

контекстно-свободен. Его можно разрешить с $L(n) = \log n$ на односторонней машине Тьюринга, так как память $\log n$ можно использовать для подсчета числа символов a и сравнения этого числа с числом символов b .

Теорема 2.

(1) $(L(n) = \log n)$ -разрешимый на двусторонней машине Тьюринга контекстно-свободный язык

$$L_3 = \{a^n b^n \mid n = 1, 2, \dots\}$$

нельзя разрешить с

$$L(n) < \cdot \log n.$$

(2) $(L(n) = n)$ -разрешимый на двусторонней магазинной машине контекстно-свободный язык L_3 нельзя разрешить с

$$L(n) < \cdot n.$$

(3) Контекстно-свободный язык

$$L_6 = \overset{\rightarrow}{\omega} s \overset{\leftarrow}{\omega},$$

где $\overset{\rightarrow}{\omega}$ — любая конечная последовательность двоичных символов, а $\overset{\leftarrow}{\omega}$ есть зеркальное изображение $\overset{\rightarrow}{\omega}$, можно разрешить на односторонней машине Тьюринга с $L(n) = n$ и нельзя разрешить с

$$L(n) < \cdot n.$$

Доказательство. Из теоремы 1 мы знаем, что L_3 $(L(n) = \log n)$ -разрешим на двусторонней машине Тьюринга. Покажем, что L_3 нельзя разрешить на двусторонней машине Тьюринга с

$$L(n) < \cdot \log n.$$

Доказательство проведем сведением к противоречию; мы покажем, что любая машина M , принимающая якобы множество последовательностей $\{a^m b^m\}$ с $L(n) < \cdot \log n$, должна также принять и последовательность $a^m b^{m+m!}$ для некоторого m . Пусть p — число, такое, что для всех натуральных чисел j

$$p^j > Q \cdot T \cdot j \cdot T^j,$$

где Q — число состояний управляющего устройства, T — длина рабочего алфавита. Так как $L(n) < \cdot \log n$, должно существовать такое число m , что $L(2m) < \log_p m$ и что M обладает менее

чем $p^{L(2m)} < t$ различными конфигурациями памяти при обработке $a^m b^m$. Если при обработке $a^m b^m$ читающая головка проходит над всей последовательностью символов b , то должны быть два b , такие, что, когда они обозреваются при этом проходе, M находится в одной и той же конфигурации. Если два b с этим свойством разделены l квадратами ленты, то, если m символов b заменить на $m + k \cdot l$ символов b , где k — любое положительное целое число, M придет на другой конец этой строки символов b в той же самой конфигурации, что и для строки из m символов b . Это справедливо для каждого прохода M над строкой из b , хотя расстояние l может быть различным для каждого прохода. Так как $m + m!$ можно для каждого $l \leq m$ записать в форме $m + k \cdot l$, при любом проходе машина не сможет различить

$$a^m b^m \quad \text{и} \quad a^m b^{m+m!}.$$

Заметим, что если машина начинает обозревать строку символов b и не доходит до ее конца, то она будет трактовать $a^m b^m$ и $a^m b^{m+m!}$ одинаковым образом. Поэтому $a^m b^{m+m!}$ должно быть принято, и мы получаем разыскиваемое противоречие.

Магазинная машина удовлетворяет, очевидно, частному варианту леммы в [1], поэтому мы опускаем подробности.

Часть 3 теоремы очевидна. Когда читающая головка односторонней машины Тьюринга пройдет через центральный маркер s , на рабочей ленте нужно запомнить информацию, достаточную для восстановления $\vec{w}$, что, очевидно, требует n квадратов.

Покажем, что каждая модель машины дает бесконечную иерархию контекстно-свободных языков (предложения АЗ, БЗ, ВЗ).

Теорема 3. *Существуют бесконечные иерархии контекстно-свободных языков для двусторонних машин Тьюринга с*

$$\log \log n \leq \cdot L(n) \leq \cdot \log n,$$

для односторонних машин Тьюринга с

$$\log n \leq \cdot L(n) \leq \cdot n$$

и для двусторонних магазинных машин с

$$\log \log n \leq \cdot L(n) \leq \cdot n.$$

Доказательство. Чтобы доказать, что существует бесконечно много классов сложности контекстно-свободных языков для двусторонних машин Тьюринга, покажем, что для всякого

рационального числа $p/q \geq 1$ существует контекстно-свободный язык, который можно разрешить с

$$L(n) = [\log \log n]^{p/q}$$

и нельзя разрешить с

$$L(n) < \cdot [\log \log n]^{p/q}.$$

Мы дадим доказательство для случая $p/q=2$, указывающее основную идею общего случая.

Рассмотрим язык L_7 , дополнение к которому следующее:

$$\begin{aligned} \bar{L}_7 = \{ & \vec{b}_1^a s \vec{b}_2^a s \dots s \vec{b}_{2m}^a s s 1^{[\log \log 2m]} s \\ & 1^{[\log \log 2m]-1} s \dots s 1^2 s 1 s s s 1 s 1^2 s \\ & 1^4 s 1^8 s \dots s 1^{2^{[\log \log 2m]^2}} \}. \end{aligned}$$

Для каждого m пусть n_m — длина соответствующей последовательности в $\bar{L}_7$. Согласно рассуждению, подобному доказательству теоремы 2, эта m -я последовательность потребует $\log[2^{[\log \log 2m]^2}]$ квадратов ленты для того, чтобы машину не «обманула» такая же последовательность с некоторым числом дополнительных единиц на конце. Итак, мы получили

$$L(n_m) \cdot \geq \log[2^{[\log \log 2m]^2}] \doteq [\log \log 2m]^2.$$

Можно показать, что n_m меньше, чем $2m \log m$. Используя монотонность L , видим, что

$$\begin{aligned} L(2m \log m) & \geq L(n_m) \cdot \geq [\log \log 2m]^2 \doteq \\ & \doteq [\log \log [2m \log m]]^2 \end{aligned}$$

или

$$L(n) \cdot \geq [\log \log n]^2.$$

Из предыдущего рассмотрения $\bar{L}_2$ ясно, что L_7 можно разрешить с

$$L(n) \leq \cdot [\log \log n]^2,$$

чем устанавливается, что $L(n) \doteq [\log \log n]^2$.

Остается только показать, что L_7 — контекстно-свободный язык. Используем метод теоремы 1; $\bar{L}_7$ представим как пересечение нескольких контекстно-свободных языков, разрешимых

односторонними магазинными машинами. В частности, $\bar{L}_7$ — пересечение следующих семи языков:

$$\begin{aligned} L_a &= \vec{b}_1^a s (\vec{b}_j^a s \vec{b}_{j+1}^a s)^* (0 + 1 + u + z)^* ss(1 + s)^*, \\ L_b &= (\vec{b}_k^a s \vec{b}_{k+1}^a s)^* ss(1 + s)^*, \\ L_c &= (0 + 1 + s + u + z)^* (0 + 1 + s) (u + z)^q ss \\ &\quad 1^q s (1^l s 1^{l-1} s)^* 1^* sss(1 + s)^*, \\ L_d &= (0 + 1 + s + u + z)^* (u + z) ss (1^p s 1^{p-1} s)^* 1^* sss(1 + s)^*, \\ L_e &= (0 + 1 + s + u + z)^* ss \left(1 + \frac{s}{2}\right)^r sss(1^* s)^r, \end{aligned}$$

где $\left(1 + \frac{s}{2}\right)^r sss(1^* s)^r$ означает произвольную строку из единиц и символов s , за которыми следует sss , за которыми в свою очередь следует r групп $1^* s$, при этом r равно числу единиц плюс половина числа символов s , содержащихся в этой произвольной строке,

$$\begin{aligned} L_f &= (0 + 1 + s + u + z)^* sss 1 s (1^t s 1^{2t} s)^* 1^* s, \\ L_g &= (0 + 1 + s + u + z)^* sss (1^v s 1^{2v} s)^*. \end{aligned}$$

Как в доказательстве теоремы 1, L_a и L_b дают начальную часть $\bar{L}_7$ до первого появления ss . Так как $\vec{b}_{2m}^a$ последовательно пронумеровано, прямо перед ss будут находиться $[\log \log 2m]$ символов u и z . Язык L_c тогда даст также $[\log \log 2m]$ единиц сразу после ss . Языки L_c и L_d образуют $\bar{L}_7$ между ss и sss . В этой части число единиц равно

$$\begin{aligned} [\log \log 2m] + ([\log \log 2m] - 1) + \dots + 2 + 1 = \\ = \frac{1}{2} \{[\log \log 2m]^2 - [\log \log 2m]\}. \end{aligned}$$

Если к этому числу прибавить половину числа всех маркеров s , то получим точно

$$[\log \log 2m]^2.$$

L_e обеспечивает в точности это число, т. е. $[\log \log 2m]^2$ маркеров s после sss . Языки L_f и L_g дадут часть последовательности после sss . Осталось только показать, что все семь языков действительно разрешимы односторонней магазинной машиной. Это относительно просто.

Подобный метод дает все рациональные степени, превосходящие единицу, $\log \log 2m$ и все рациональные дробные (не

большие чем 1) степени $\log 2t$ и многих других функций. Можно пояснить, почему иерархия для двусторонних машин Тьюринга при этом методе ограничена $L(n) = \log n$. Чтобы доказать, что требуется $L(n)$ квадратов, мы конструировали последовательность, имеющую $2^{L(n)}$ единиц на конце. Но это число единиц должно быть небольшим по сравнению с n — длиной последовательности, т. е.

$$2^{L(n)} \leq n$$

или

$$L(n) \leq \log n.$$

Поэтому нужно использовать другие методы для разыскивания контекстно-свободных языков, требующих $L(n) \cdot > \log n$.

Аналогичным образом можно доказать теорему для двух других моделей машин.

Как говорилось выше, мы не умеем доказывать (и предполагаем, что это невозможно), что все контекстно-свободные языки можно разрешать с $L(n) = \log n$. Однако мы укажем алгоритм, который разрешает все контекстно-свободные языки и требует $L(n) = [\log n]^2$.

Теорема 4. *Все контекстно-свободные языки можно разрешить на двусторонней машине Тьюринга с $L(n) = [\log n]^2$.*

Доказательство. Укажем нужный алгоритм. Опять для краткости дадим неформальное описание.

Предположим для простоты изложения, что все правила языка содержат только два символа (т. е. $A \rightarrow BC$ или $A \rightarrow aB$). Нам потребуется вспомогательная лемма. Рассмотрим в таком языке слово из n символов. Мы покажем, что в этом слове должна быть по крайней мере одна последовательность символов длины l

$$n/3 \leq l \leq 2n/3,$$

такая, что она является в точности потомком одного нетерминального символа. Чтобы показать это, рассмотрим начальное правило, порождающее наше предложение:

$$S \rightarrow AB.$$

Если ни A ни B не являются нетерминальными символами с желаемым свойством, то потомок одного из них, скажем B , имеет длину, большую чем $2n/3$. Рассмотрим непосредственное потомство B ,

$$B \rightarrow CD.$$

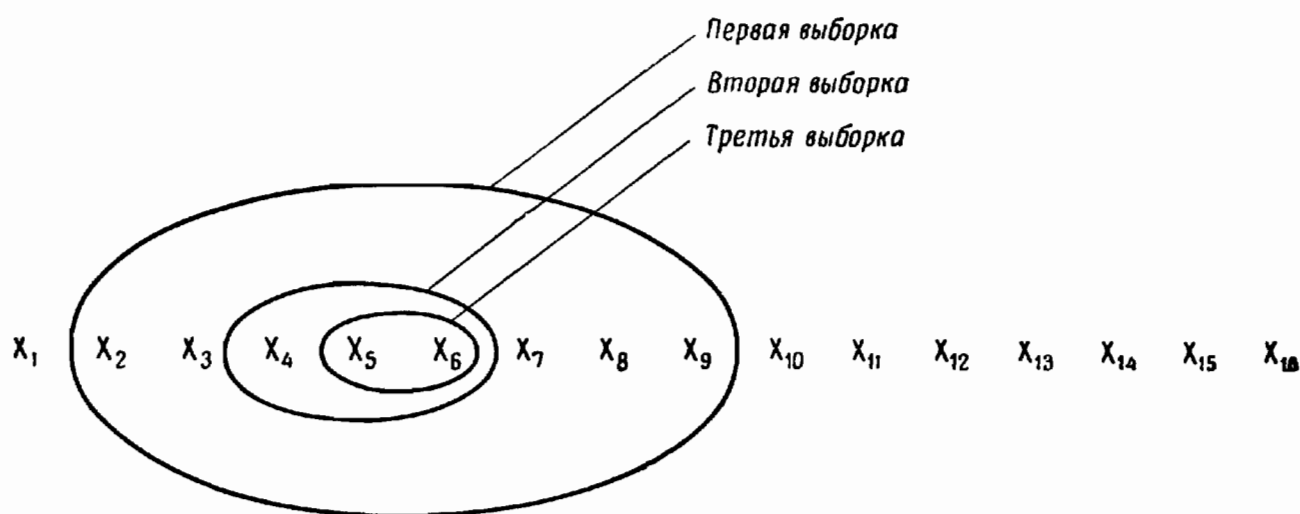
По крайней мере один из них, например D , должен обладать потомком с длиной, большей чем $1/2(2n/3) = n/3$. Итак, если C и D оба не обладают нужным свойством, то потомок D должен иметь длину, большую чем $2n/3$, и процесс можно повторять, пока в конце концов не определится нетерминальный символ с желаемым свойством. Этим завершается доказательство вспомогательной леммы.

Первый шаг разрешения состоит в том, чтобы угадать положение потомка нетерминального символа, обладающего этим свойством.

1. Из данного слова длины n систематической процедурой делаем выборку подслова с длиной l

$$n/3 \leq l \leq 2n/3.$$

Процедура выделения должна быть систематической, чтобы можно было, если первая проба неудачна, перейти к новой пробе, отличной от всех предыдущих, наиболее простым способом.



Р и с. 1. Первые четыре шага разрешающего алгоритма.

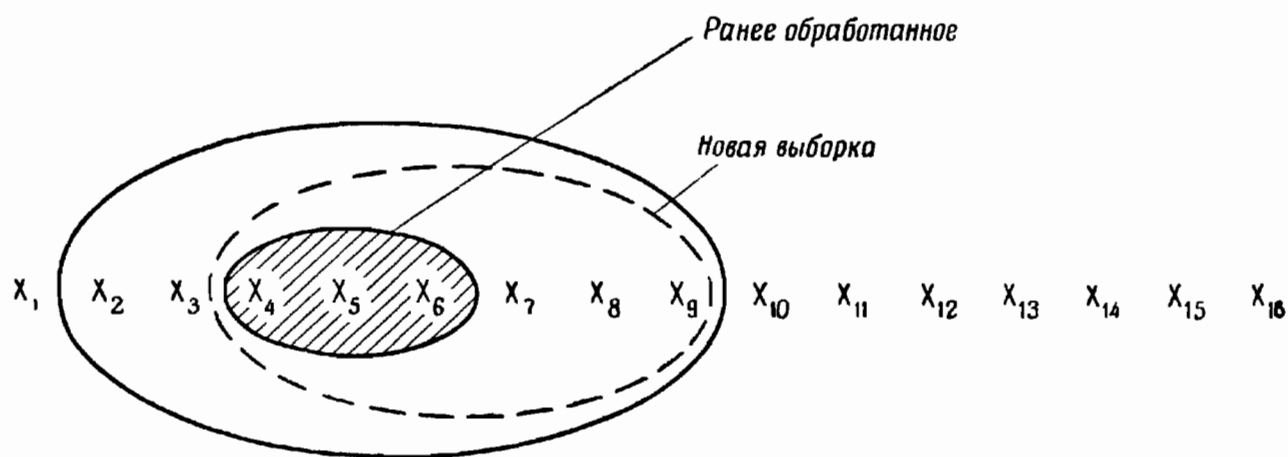
2. Память объема $(\log n)^2$ разделяется на $\log n$ блоков из $\log n$ квадратов каждый. В первом блоке запоминаются положения в данном слове (т. е. адреса) двух концевых символов выделенного подслова.

3. Две вышеописанные манипуляции производятся над выделенным подсловом, т. е. внутри его выделяется новое подслово, длина которого заключена между $1/3$ и $2/3$ длины первого подслова, и записывается его адрес во второй блок памяти.

4. Повторяем этот процесс до тех пор, пока не получим подслова, состоящего только из двух символов. Очевидно, что для всей процедуры потребуется не более чем $\log(2n/3)$ блоков памяти.

Первые четыре шага этого процесса показаны схематически на рис. 1.

5. Теперь можно определить, соответствует ли последняя выборка какому-либо правилу грамматики. Если нет, то производится новая последняя выборка. Если невозможна последняя выборка, соответствующая какому-либо правилу, то производится нужная предпоследняя выборка и снова подыскивается нужная последняя выборка. Если ни одна из предпоследних выборок не подходит, то изменяем предпредпоследнюю выборку и повторяем весь процесс, пока не найдем последнюю выборку, которая соответствует какому-либо правилу. (Если при исчерпании всех выборок не получили такой ситуации, то ясно, что слово не входит в язык.)



Р и с. 2. Десятый шаг алгоритма.

6. Если последняя выборка соответствует некоторому правилу, то нетерминальный символ, стоящий в левой части правила, пишем в последний блок памяти.

7. Теперь можно определить, соответствует ли предпоследняя выборка какому-либо правилу. Если нет, то производим новую предпоследнюю выборку и повторяем процедуру, описанную в 5. Если все выборки второго и более высоких уровней исчерпаны, а соответствующее правило не найдено, то производим новую последнюю выборку и повторяем шаг 5.

8. Если предпоследняя выборка соответствует одному или нескольким правилам, то нетерминальные символы, стоящие в левой части правил, пишем в предпоследний блок памяти.

9. Теперь последний блок памяти не нужен, и мы его стираем.

10. Как показано на рис. 2, новая выборка подслова производится внутри третьей от конца выборки и его адрес записывается в последний блок памяти, уже годный к употреблению.

11. Процесс повторяется и отбираются все более длинные подслова. Как только подслово отобрано, все блоки памяти, используемые для хранения информации о правилах, работающих внутри подслова, становятся ненужными и очищаются, чтобы

быть готовыми пометать другие выборки, как на шаге 10. Если данное предложение входит в язык, то описанная процедура в конце концов опознает, что все предложение порождается начальным символом S .

5. СРАВНЕНИЕ ИЕРАРХИЙ КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ

Нужно подчеркнуть, что иерархии контекстно-свободных языков, определяемые четырьмя моделями машин, различны. Языки, легко разрешимые на машинах одного типа, могут быть трудно разрешимыми машинами другого типа, и наоборот.

Наиболее удивительный результат подобного рода относится к односторонним магазинным машинам и двусторонним машинам Тьюринга. Любой нерегулярный контекстно-свободный язык, который можно вообще разрешить на односторонней магазинной машине, требует по крайней мере $L(n) \doteq \log n$ на двусторонней машине Тьюринга. Поэтому всю бесконечную иерархию контекстно-свободных языков, «легко» разрешимых двусторонней машиной Тьюринга, т. е. для которых

$$\log \log n \leqslant L(n) < \log n,$$

невозможно разрешать односторонними магазинными машинами. А все те контекстно-свободные языки, которые можно разрешать на односторонней магазинной машине, «трудны» для двусторонней машины Тьюринга, т. е. требуют по крайней мере $L(n) \doteq \log n$. Об этом более точно говорит следующая теорема.

Теорема 6. *Все нерегулярные контекстно-свободные языки, которые можно разрешить на односторонней магазинной машине, требуют по крайней мере $L(n) \doteq \log n$ для своего разрешения на двусторонней машине Тьюринга.*

Доказательство. В литературе имеются по крайней мере три разных определения односторонних магазинных машин:

(1) Машина, которая может печатать или стирать только по одному символу на ленте при каждом вводном символе, при этом магазинная лента должна быть пуста, если вводное слово принимается.

(2) Машина, подобная (1), с тем исключением, что не требуется пустоты ленты при разрешении.

(3) Машина, подобная (2), с тем исключением, что при каждом вводном символе машина может печатать произвольное число символов на магазинной ленте.

Теорема верна для всех трех типов машин. (Тип (3), наиболее общий, определен в [1].) Однако даже набросок доказа-

тельства для типа (3) слишком громоздок. Поэтому мы дадим набросок доказательства для машины типа (2).

Основная идея доказательства состоит в следующем. Мы сначала покажем, что если нерегулярный язык разрешим на односторонней магазинной машине, то существуют слова, требующие для своей записи довольно длинной магазинной ленты, которая в дальнейшем должна быть прочитана. Затем мы покажем, что каждое такое слово должно состоять из

(1) некоторого слова (назовем его w_A), которое повторяется достаточно много раз, заставляя магазинную ленту при записи его возрастать по длине;

(2) некоторого другого слова (назовем его w_B), которое повторяется столько же раз, сколько и w_A , заставляя печатать что-нибудь на ленте. Разрешение, определяемое равенством числа вхождений w_A и w_B , согласно теореме 2, потребует $L(n) \doteq \log n$ на двусторонней машине Тьюринга.

Теперь дадим детали доказательства.

Если нерегулярный язык разрешим на односторонней магазинной машине, то существуют слова, требующие довольно большой магазинной ленты. Более точно, для всякого числа l существует слово w_l , при обработке которого (как начала вводного слова) на ленте остается некоторое число l_0 символов и $l_0 \geq l$; кроме того, всякое другое слово, скажем w' , использующее менее чем l_0 квадратов ленты, можно отличить от w_l . То есть существует некоторое другое слово w_1 , такое, что два предложения

$$S_1 = w_l w_1,$$

$$S_2 = w' w_1$$

обладают следующим свойством: либо S_1 входит в язык и S_2 не входит, либо наоборот.

Так как l — произвольное число, выберем его равным

$$l = QIT + Q^2 I^2 T + 2,$$

где Q — число состояний управляющего устройства машины, а I и T — соответственно длины вводного и рабочего алфавита. $QIT + 1$ квадрат в начале ленты (те, что заполняются в первую очередь) назовем дном ленты. Оставшиеся квадраты назовем верхом ленты.

Покажем сначала, что для этого l и соответствующего w_l существует некоторое w' , требующее меньшего числа квадратов, но из-за которого магазинная машина должна посетить дно магазинной ленты для w_l , чтобы отличить w' от w_l .

Так как дно ленты для w_l состоит из $QIT + 1$ квадратов, оно должно содержать два квадрата, скажем t_1 и t_2 , такие, что после переработки w_l

- (а) в t_1 и t_2 напечатаны одинаковые символы;
 (б) когда эти символы печатались, машина находилась в одном и том же состоянии и соответствующие вводные символы (в w_l) были одинаковы.

Тогда w_l можно записать как

$$w_l = w_\alpha s_1 w_\beta s_1 w_\gamma,$$

где оба вхождения s_1 выделяют символы, которые заставили (последний раз) машину печатать в квадратах t_1 и t_2 . Теперь рассмотрим w'

$$w' = w_\alpha s_1 w_\gamma.$$

При обработке этого слова получим в точности такую же магазинную ленту, что и при w_l , с тем исключением, что будет выкинут кусок между t_1 и t_2 (включая t_2). Так как ленты для w' и w_l отличаются только на дне, то, чтобы отличить

$$S_1 = w_l w_1$$

от

$$S_2 = w' w_1,$$

машина должна посетить дно ленты. Итак, при обработке S_1 , к примеру, машина должна сначала напечатать всю ленту, дно и верх, а затем должна «стереть» весь верх, чтобы прочитать дно.

Теперь рассмотрим верх ленты, получаемой при обработке S_1 . Так как w_l использует $l_0 > l = QIT + Q^2I^2T + 2$ квадратов, то верх этой ленты состоит по крайней мере из $Q^2I^2T + 1$ квадратов. Следовательно, он содержит по крайней мере два таких квадрата, скажем t_3 и t_4 , что при переработке S_1

- (а) в t_3 и t_4 печатались (в последний раз) одни и те же символы;

- (б) когда в t_3 и t_4 печатались (в последний раз) эти символы и когда (позже) они прочитывались, (все четыре раза) машина находилась в одном и том же состоянии, а соответствующие вводные символы были одинаковы.

Поэтому S_1 можно записать так:

$$S_1 = w_4 s_2 w_5 s_2 w_6 s_2 w_7 s_2 w_8,$$

где первые два вхождения s_2 выделяют символы, которые заставили (последний раз) печатать в t_3 и t_4 , а последние два вхождения s_2 выделяют символы, когда t_3 и t_4 были прочтены.

Теперь посмотрим, что произойдет при обработке слова вида

$$\mathcal{P}_1 = w_4 s_2 (w_5 s_2)^n w_6 s_2 (w_7 s_2)^n w_8.$$

Вычисления очень похожи на вычисления при S_1 . Дополнительные вводные символы в $(w_5 s_2)^n$ дадут дополнительный кусок

ленты, совершенно одинаковый с куском первоначальной ленты между t_4 и t_3 . Если лента для S_1 имеет вид

$$\tau_1 t \tau_2 t \tau_3$$

(где t — символ, напечатанный в t_3 и t_4), то лента для $\mathcal{P}_1$ будет иметь вид

$$\tau_1 t (\tau_2 t)^n \tau_3.$$

Однако вводные символы в $(w_7 s_2)^n$ заставят стереть эти дополнительные квадраты ленты (как раньше для S_1). Поэтому результаты вычислений одинаковы и все слова вида $\mathcal{P}_1$ принимаются (или отвергаются), если принимается (или отвергается) слово S_1 .

Когда машина обозревает верх ленты, она не может отличить S_2 от S_1 . Поэтому квадраты t_3 и t_4 обладают теми же свойствами при обработке S_2 , что и при S_1 . В частности, S_2 можно записать как

$$S_2 = w_9 s_2 w_5 s_2 w_6 s_2 w_7 s_2 w_{10},$$

где w_9 и w_{10} появляются из-за другого дна ленты, а s_2 , w_5 , w_6 , w_7 те же, что и для S_1 . Получаем также, что множество слов

$$\mathcal{P}_2 = w_9 s_2 (w_5 s_2)^n w_6 s_2 (w_7 s_2)^n w_{10}$$

принимается или отвергается соответственно тому, принимается или отвергается S_2 .

Итак, мы показали, что либо $\mathcal{P}_1$ содержится в языке, допускаемом машиной, а $\mathcal{P}_2$ не содержится, либо наоборот. Рассмотрим теперь два множества предложений

$$\mathcal{P}'_1 = w_4 s_2 (w_5 s_2)^{n_1} w_6 s_2 (w_7 s_2)^{n_2} w_8,$$

$$\mathcal{P}'_2 = w_9 s_2 (w_5 s_2)^{n_1} w_6 s_2 (w_7 s_2)^{n_2} w_{10},$$

где $(n_2 - n_1) \geq K$ и K достаточно велико, чтобы при обработке $\mathcal{P}'_1$ и $\mathcal{P}'_2$ не стирался весь верх ленты (вспомним, что мы рассматриваем только те машины, которые для каждого вводного символа могут стереть только один символ на ленте). Так как машина не читает дно ленты при обработке $\mathcal{P}'_1$ и $\mathcal{P}'_2$, то она не может отличить их и они оба принимаются либо оба отвергаются.

Пусть, например, язык таков, что $\mathcal{P}_1$ содержится в языке, а $\mathcal{P}_2$, $\mathcal{P}'_1$, $\mathcal{P}'_2$ не содержатся. Если машина Тьюринга должна разрешать этот язык и отличать $\mathcal{P}_1$ от $\mathcal{P}_2$, то она должна подсчитать число $(n_2 - n_1)$ и сравнить его с K . Тогда по той же причине, что и в теореме 2, она должна на самом деле уметь считать до n ,

что требует по крайней мере $L(n) \doteq \log n$ квадратов ленты. Аналогично рассуждаем в остальных случаях.

Теорема 6 гласит, что требуется по крайней мере $L(n) \doteq \doteq \log n$ квадратов ленты на двусторонней машине Тьюринга, чтобы промоделировать одностороннюю магазинную машину, производящую нерегулярное разрешение. Теорема 5 гласит, что для этого требуется самое большее $L(n) \doteq (\log n)^2$, даже если магазинная машина недетерминирована. Ранее было замечено, что вопрос, действительно ли требуется $(\log n)^2$, остается открытым. Мы предполагаем, что это так даже для детерминированных магазинных машин.

Другой открытый вопрос относится к аналогичным результатам для двусторонних магазинных машин; какая функция $L(n)$ потребуется, чтобы на двусторонней машине Тьюринга промоделировать двустороннюю магазинную машину, производящую нерегулярное разрешение. Здесь результат, быть может, зависит от $L(n)$ для магазинной машины.

ЛИТЕРАТУРА

1. Stearns R. E., Hartmanis J., Lewis II P. M., Hierarchies of memory limited computations, IEEE Conf. Rec. Switch. Circuit Theory and Logic. Design, 1965, 179—190. (Настоящий сборник, стр. 301—319.)
2. Chomsky N., Three models for the description of language, *IRE Trans. on Information Theory*, IT-2 (1956), 113—124. (Русский перевод: Кибернетический сборник, вып. 2, 1961.)
3. Landweber P. S., Three theorems on phrase structure grammars of Type I, *Information and Control*, 6 (1963), 131—137.
4. Kuroda S. Y., Classes of languages and linear-bounded automata, *Information and Control*, 7 (1964), 207—223.
5. Chomsky N., Formal properties of grammars, Chapter 12 of Handbook of Mathematical Psychology, v. II, 1963. (Русский перевод: Кибернетический сборник (новая серия), вып. 2, «Мир», М., 1966, 121—230.)

ОБ ОБЪЕМЕ ПАМЯТИ, НЕОБХОДИМОМ ДЛЯ РАСПОЗНАВАНИЯ БЕСКОНТЕКСТНЫХ ЯЗЫКОВ ¹⁾

Дж. Хартманис

ВВЕДЕНИЕ

В работе исследуется распознавание бесконтекстных языков машинами Тьюринга. Мы ограничимся машинами Тьюринга с двумя лентами: лентой входа с двусторонним движением со считыванием, но без записи и рабочей лентой с двусторонним движением со считыванием и с записью.

Определение. Множество последовательностей W называется *распознаваемым с памятью $L(n)$* , коль скоро существует машина Тьюринга $\mathcal{J}$, которая, во-первых, распознает W (т. е. останавливается на всех входных последовательностях и при этом допускает цепочки из W и отвергает цепочки, не принадлежащие W) и, во-вторых, использует не более $L(n)$ квадратов на рабочей ленте при оперировании с входом длины n .

В [1] было показано следующее:

(1) Каждый бесконтекстный язык распознаваем с памятью не более $L(n) = [\log n]^2$.

(2) Существует нерегулярный бесконтекстный язык, который можно распознавать, используя всего лишь $L(n) = [\log \log n]$ квадратов ленты. Но если язык распознаваем с памятью $L(n)$ и

$$\lim_{n \rightarrow \infty} \frac{L(n)}{\log \log n} = 0,$$

то язык регулярен (т. е. распознаваем при помощи устройства с конечной памятью).

(3) Бесконтекстный язык $\{0^n 1^n \mid n = 1, 2, \dots\}$ требует для своего распознавания объем памяти, равный $[\log n]$, т. е. если

$$\inf_{n \rightarrow \infty} \frac{L(n)}{\log n} = 0,$$

то язык $\{0^n 1^n\}$ не может быть распознаваем с памятью $L(n)$.

(4) Существует неограниченно много классов сложности (относительно памяти) бесконтекстных языков с отличными друг

¹⁾ Hartmanis J., On memory requirements for context-free language recognition, *J. ACM*, 14, № 4 (1967), 663—665.

от друга функциями памяти $L(n)$, расположенными между $[\log \log n]$ и $[\log n]$. (Неизвестно, требуется ли для распознавания какого-либо бесконтекстного языка объем памяти более чем $[\log n]$.)

Таким образом, доказано, что распознавание нерегулярных бесконтекстных языков может требовать резко различных объемов памяти, которые занимают экспоненциальную шкалу от $L(n) = [\log \log n]$ до $L(n) = [\log n]$. Далее, известно, что невозможно эффективно узнать, когда бесконтекстная грамматика порождает регулярный язык [2], и, следовательно, неразрешим вопрос о том, распознаваем ли некоторый язык с конечной длиной ленты.

Ниже доказывается, что если даже заранее известно, что язык нерегулярен, то вопрос о длине ленты, необходимой для его распознавания, остается алгоритмически неразрешимым. Как и многие другие доказательства неразрешимости различных свойств бесконтекстных языков, это доказательство опирается на неразрешимость комбинаторной проблемы Поста [2, 3]¹⁾.

ТЕОРЕМА О НЕРАЗРЕШИМОСТИ

Вначале построим два хорошо известных вспомогательных языка, используемых в доказательстве теоремы о неразрешимости. Пусть A и B — два списка, т. е. два упорядоченных набора из k непустых бинарных последовательностей

$$A = (a_1, a_2, \dots, a_k), \quad B = (b_1, b_2, \dots, b_k),$$

где $k \geq 1$, a_i, b_i из $\{0, 1\}^*$, и пусть i означает двоичную запись числа i . Тогда полагаем по определению

$$L(A) = \{i_1 \otimes i_2 \otimes \dots \otimes i_p \otimes \otimes a_{i_p} \dots a_{i_2} a_{i_1}\}$$

и аналогично

$$L(B) = \{j_1 \otimes j_2 \otimes \dots \otimes j_q \otimes \otimes b_{j_q} \dots b_{j_2} b_{j_1}\}.$$

Языки $L(A)$ и $L(B)$ могут распознаваться детерминированным автоматом PDS и, следовательно, являются бесконтекстными языками; их дополнения $\overline{L(A)}$ и $\overline{L(B)}$ также бесконтекстны [2].

Лемма 1. Если

$$L(A) \cap L(B) \neq \emptyset,$$

¹⁾ См. также Марков А. А., Теория алгорифмов, Труды МИАН, т. 42, гл. VI, § 9, 1954. и Мальцев А. И., Алгоритмы и рекурсивные функции, М., 1965, стр. 301. — Прим. перев.

то для распознавания этого пересечения требуется длина ленты $L(n) = \lceil \log n \rceil$.

Доказательство. Если $L(A) \cap L(B) \neq \emptyset$, то существует последовательность

$$i_1 * i_2 * \dots * i_r * a_{i_r} \dots a_{i_2} a_{i_1},$$

принадлежащая $L(A) \cap L(B)$. Далее, последовательность

$$(i_1 * i_2 * \dots * i_r)^p * (a_{i_r} \dots a_{i_2} a_{i_1})^q$$

принадлежит $L(A) \cap L(B)$ в том и только в том случае, если $p = q$. Допустим теперь, что у нас имеется машина Тьюринга $\mathcal{J}$, распознающая непустое множество

$$L(A) \cap L(B)$$

и работающая с памятью $L(n)$. Тогда, поместив перед $\mathcal{J}$ устройство с конечным числом состояний, отвергающее все входные цепочки, не принадлежащие множеству

$$\{(i_1 * i_2 * \dots * i_r)^p * (a_{i_r} \dots a_{i_2} a_{i_1})^q \mid p = 1, 2, \dots, \\ q = 1, 2, \dots\},$$

мы превратим машину $\mathcal{J}$ в другую машину $\mathcal{J}'$, которая распознает множество

$$W = \{(i_1 * i_2 * \dots * i_r)^N * (a_{i_r} \dots a_{i_2} a_{i_1})^N \mid N = 1, 2, \dots\}.$$

Поскольку собственная программа $\mathcal{J}$ никак не изменилась при переходе к $\mathcal{J}'$, то отсюда следует, что $\mathcal{J}'$ использует не больше ленты, чем $\mathcal{J}$. С другой стороны, любое устройство, распознающее множество W , должно уметь считать до N , а поскольку N пропорционально длине входной последовательности, мы можем заключить (из рассуждений, аналогичных тем, которые проводились в [1] для доказательства утверждения (3) из введения), что распознавание W требует памяти $L(n) = \lceil \log n \rceil$. Доказательство закончено.

Лемма 2. Задача: верно ли, что $L(A) \cap L(B) = \emptyset$, алгоритмически неразрешима.

Доказательство. Множество $L(A) \cap L(B)$ непусто тогда и только тогда, когда существует последовательность индексов $i_1, i_2, \dots, i_r$, такая, что $a_{i_1} a_{i_2} \dots a_{i_r} = b_{i_1} b_{i_2} \dots b_{i_r}$. Таким образом, $L(A) \cap L(B) \neq \emptyset$ тогда и только тогда, когда комбинаторная проблема Поста [3] имеет положительное решение для списков A и B . А поскольку известно, что комбинаторная

проблема Поста алгоритмически неразрешима, то неразрешима и проблема пустоты пересечения $L(A) \cap L(B)$.

Теперь воспользуемся этим результатом, чтобы доказать неразрешимость вопроса о длине ленты, необходимой для распознавания нерегулярных бесконтекстных языков.

Теорема. *Невозможно эффективно решить по любой бесконтекстной грамматике, порождающей нерегулярный язык, распознаваем ли ее язык $L(G)$ с памятью $L(n) = [\log \log n]$.*

Доказательство. Прежде всего отметим, что $\overline{L(A)}$ и $\overline{L(B)}$ — бесконтекстные языки, и в силу этого язык

$$\overline{L(A) \cap L(B)} = \overline{L(A)} \cup \overline{L(B)}$$

также является бесконтекстным языком. Поскольку распознавание множеств

$$\overline{L(A) \cap L(B)} \quad \text{и} \quad L(A) \cap L(B)$$

требует одной и той же длины ленты, то наш бесконтекстный язык $\overline{L(A)} \cup \overline{L(B)}$ либо представляет собой множество всех цепочек (в терминальном алфавите), если $L(A) \cap L(B) = \emptyset$, и в этом случае регулярен, либо является нерегулярным языком, требующим для своего распознавания ленту длины $[\log n]$. В силу леммы 2 вопрос о том, какой из этих двух случаев имеет место, алгоритмически неразрешим.

Во-вторых, рассмотрим множество цепочек

$$W = \{1 \times 2^T \times 3 \times 4^T \times \dots \times (2n)^T \mid n = 1, 2, \dots\},$$

где i — двоичная запись числа i , а $(i)^T$ означает обращенное i (т. е. слово i , записанное в обратном порядке). Легко показать, что W и $\overline{W}$ — нерегулярные множества и что оба они распознаваемы с памятью $[\log \log n]$; в [1] было доказано, что $\overline{W}$ — бесконтекстный язык. Следовательно,

$$\mathcal{L} = \overline{W} \dagger [\overline{L(A)} \cup \overline{L(B)}],$$

где $\dagger$ — новый знак алфавита¹⁾ — представляет собой нерегулярный бесконтекстный язык, распознавание которого требует ленты $[\log n]$, если $L(A) \cap L(B) \neq \emptyset$, и $[\log \log n]$, если $L(A) \cap L(B) = \emptyset$.

Отсюда можно заключить, что для нерегулярных бесконтекстных языков алгоритмически неразрешим вопрос: можно ли обойтись при распознавании минимальной длиной ленты $L(n) =$

¹⁾ Очевидно, имеется в виду, что

$$\mathcal{L} = \{x \dagger y \mid x \in \overline{W}, y \in [\overline{L(A)} \cup \overline{L(B)}]\}.$$

$= \lceil \log \log n \rceil$ или потребуется бо́льшая ее длина? Иначе говоря, неразрешим вопрос: как быстро растет длина используемой рабочей ленты при растущей длине входной цепочки в процессе распознавания нерегулярных бесконтекстных языков?

ЛИТЕРАТУРА

1. Lewis P. M., Hartmanis J., Stearns R. E., Memory bounds for recognition of context-free and context-sensitive languages, IEEE Conf. Record on Switching Circuit Theory and Logical Design, IEEE Pub. 16C13, 1965, 192—202. (Настоящий сборник, стр. 320—338.)
2. Ginsburg S., The mathematical theory of context-free languages, McGraw-Hill, N. Y., 1966. (Русский перевод: Гинзбург С., Математическая теория контекстно-свободных языков, «Мир», 1970.)
3. Post E. L., A variant of a recursively unsolvable problem, *Bull. AMS*, 52 (1946), 264—268.

РАСПОЗНАВАНИЕ И АНАЛИЗ КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ ЗА ВРЕМЯ n^3 ¹⁾

Д. Х. Янгер

СПИСОК ОБОЗНАЧЕНИЙ

$\rightarrow$	„может быть переписано, как“
$\Rightarrow$	„порождает“
T	терминальный словарь
N	нетерминальный словарь
S	специальный нетерминальный символ
L	язык
G	грамматика
φ, ψ	цепочки в $N \cup T$
$S_t = s_1 \dots s_n$	проверяемая цепочка
R	матрица распознавания с элементами $r(i, j, k)$
M	матрица неоднозначности с элементами $m(i, j, k)$
P	матрица анализа с элементами $p(i, j, k)$
G_t	матрица терминальных правил грамматики с элементами $g_t(k, l)$
G_c	матрица двойных правил грамматики с элементами $g_c(k, k_1, k_2)$
i, t	длины подцепочек в S_t
j	номер первого символа данной подцепочки в S_t
k, k_1, k_2	индексы над множеством нетерминальных символов
l	индекс над множеством терминальных символов
n	длина S_t
q	число нетерминальных символов
s	число терминальных символов.

1. ВВЕДЕНИЕ

При выборе языка программирования для ЭЦВМ важным критерием является простота проверки правильности построения данного предложения. В этой статье предлагается алгоритм распознавания, который применим к любому контекстно-свободному языку; этот алгоритм особенно хорош тем, что выполняется за число шагов, пропорциональное кубу числа символов проверяе-

¹⁾ Younger D. H., Recognition and parsing of context-free languages in time n^3 , *Information and Control*, 10, № 2 (1967), 189—208.

мой цепочки. Такая эффективность стимулирует дальнейшие поиски контекстно-свободных языков, пригодных для практического программирования.

Кроме того, алгоритм включает детальный грамматический анализ проверяемой цепочки. В качестве побочного продукта этого анализа можно получить много дополнительных сведений о грамматической структуре цепочки. При небольшом изменении алгоритма можно с его помощью вычислять степень неоднозначности предложения, т. е. число различных способов порождения его грамматикой. С другой стороны, если некоторая цепочка является предложением, производя с ней некоторые дополнительные операции (число которых не превышает куб длины предложения), мы получаем полное описание грамматической структуры данного предложения — матрицу анализа.

В качестве развития этих понятий в работе устанавливается связь между алгоритмом распознавания и классификацией языков, основанной на числе шагов, требуемых для распознавания на машине Тьюринга. Читателя, возможно, удивит, что даже на такой простой модели вычислительной машины, как машина Тьюринга¹⁾, для выполнения нашего алгоритма требуется число шагов, пропорциональное лишь кубу длины цепочки.

В проблематике, связанной с исследованием естественных языков, предложены многочисленные алгоритмы анализа; из них автору известны предсказывающий анализатор [9, 10], алгоритм анализа Кока [6], алгоритм анализа Сакаи [13] и алгоритм анализа Кея [8]; последний стоит ближе всего к алгоритму, приводимому в данной статье. Ни для одного из этих алгоритмов не даны определенные оценки времени анализа. Обзор сравнительной эффективности, произведенный для предложенных алгоритмов распознавания контекстно-свободных языков на основании опытных данных, приведен в [4]²⁾.

С другой стороны, в то время как алгоритмы, предлагаемые в этой статье, претендуют на минимальность в смысле используемого времени, алгоритм из [12] замечателен тем, что при длине n входной цепочки длина рабочей зоны пропорциональна $(\log_2 n)^2$, т. е. меньше, чем у всех известных алгоритмов.

Прежде чем описывать алгоритм, мы сделаем обзор основных понятий математической лингвистики (введение в математическую лингвистику см. в [1]).

¹⁾ Имеется в виду многоленточная машина Тьюринга. Для одноленточных машин Тьюринга в статье Танигути и Касами (Taniguchi K., Kasami T., *Inform. and Control*, 14, № 3 (1969), 278—284) получена верхняя оценка времени распознавания порядка n^4 , где n — длина проверяемой цепочки. — *Прим. ред.*

²⁾ Касами [11] независимо показал, что контекстно-свободные языки распознаваемы за время n^3 .

2. ОСНОВНЫЕ ПОНЯТИЯ КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ

Словарь есть данное множество символов, состоящее из двух непересекающихся подмножеств T и N , называемых соответственно **терминальными** и **нетерминальными** словарями. Элементы T (терминальные символы) обозначаются строчными буквами или цифрами, например $s, 0, 1$. Нетерминальные символы обозначаются заглавными буквами, например A, B .

N содержит специальный символ, обычно обозначаемый буквой S . **Цепочка в NUT** есть конечная последовательность символов NUT . **Длина** цепочки — число содержащихся в ней элементов. **Язык L** есть произвольное множество цепочек в T ; каждая цепочка, содержащаяся в L , называется **предложением** языка L .

Грамматика G есть конечное множество правил, которые рекурсивно определяют предложения языка. В **контекстно-свободной** грамматике все **правила** имеют вид $A \rightarrow \psi$, где A — нетерминальный символ, ψ — непустая цепочка в NUT . С помощью правила $A \rightarrow \psi$ цепочка в NUT вида $\phi A \phi'$ **порождает** цепочку $\phi \psi \phi'$, где ϕ и ϕ' — цепочки в NUT . Если в последовательности $\phi_1, \dots, \phi_n$ цепочек в NUT ϕ_i порождает ϕ_{i+1} согласно некоторому правилу из G для каждого i между 1 и $n-1$, то говорят, что ϕ_1 **порождает** ϕ_n с помощью G , и пишут $\phi_1 \Rightarrow \phi_n$. **Язык $L(G)$** , порожденный G , есть множество цепочек в T , которые порождены S — цепочкой, состоящей из единственного специального нетерминального символа. Язык, порожденный контекстно-свободной грамматикой, называется **контекстно-свободным языком**.

Если все правила грамматики G имеют общий вид $A \rightarrow BC$ или $A \rightarrow a$, где A, B, C — произвольные нетерминальные символы, a — произвольный терминальный символ, то говорят, что G — **нормальная грамматика**. В такой грамматике правила вида $A \rightarrow a$ называют **терминальными**, а правила вида $A \rightarrow BC$ — **двойными правилами**. Данную контекстно-свободную грамматику G легко привести к нормальной грамматике G' , такой, что $L(G) = L(G')$. Мы приведем пример такой редукции; общее доказательство имеется в [2]:

G :

$$\begin{aligned} S &\rightarrow SBS, \\ S &\rightarrow s, \\ B &\rightarrow BB, \\ B &\rightarrow BS, \\ B &\rightarrow 0, \\ B &\rightarrow 1. \end{aligned} \tag{1}$$

Все правила этой грамматики, за исключением первого, уже имеют требуемый вид. Первое правило заменяем двумя

$$S \rightarrow SA,$$

$$A \rightarrow BS,$$

где A — новый нетерминальный символ; таким образом, получаем нормальную грамматику G' :

Двойные правила	Терминальные правила	
$S \rightarrow SA,$	$S \rightarrow s,$	
$A \rightarrow BS,$	$B \rightarrow 0,$	(2)
$B \rightarrow BB,$	$B \rightarrow 1.$	
$B \rightarrow BS,$		

За исключением правил $S \rightarrow SA$ и $A \rightarrow BS$, образующих представление правила $S \rightarrow SBS$ в нормальной форме, нормальная грамматика существенно не отличается от контекстно-свободной грамматики, из которой она построена.

3. МАТРИЦА РАСПОЗНАВАНИЯ

Пусть дана контекстно-свободная грамматика, и пусть она приведена к нормальной форме. Пусть эта нормальная грамматика задана в виде двух множеств: множества упорядоченных пар, представляющих терминальные правила, и множества упорядоченных троек, представляющих двойные правила. Дана также цепочка символов в терминальном словаре языка. Необходимо узнать, входит ли эта цепочка в язык, порожденный данной грамматикой.

Определение. Алгоритм *распознает* данный язык L , если для любой цепочки в терминальном словаре языка он за конечное число шагов правильно решает вопрос о том, является ли данная цепочка предложением в языке L .

Этот распознающий алгоритм будет сформулирован при помощи матрицы распознавания. Для каждой подцепочки проверяемой цепочки S_i эта матрица содержит все символы из N , порождающие эту подцепочку. В частности, в этой матрице перечислены символы, которые порождают всю цепочку S_i : если в этом списке содержится символ S , то цепочка S_i является предложением в данном языке, если S не входит в этот список, то она не является таковой. Такая матрица, кроме информации, связанной с распознаванием, содержит много дополнительных сведений о грамматической структуре данного предложения.

В первой части статьи мы будем употреблять эту матрицу только для распознавания, однако позднее мы используем ее как основу для построения матрицы анализа.

Проверяемая цепочка задана в виде

$$S_t = s_1 s_2 \dots s_n, \text{ где } s_j \in T, \quad j = 1, \dots, n.$$

Подцепочка длины i , у которой первый символ находится на j -м месте в S_t , — это цепочка $s_j s_{j+1} \dots s_{j+i-1}$.

Для наших целей будем записывать элементы нетерминального словаря нормальной грамматики¹⁾ в виде A_k , где $A_1 = S$ — специальный символ из N , а индекс k пробегает значения от 1 до q (число элементов N).

Определение. Для данной нормальной грамматики G в словаре $T \cup N$ и данной цепочки $S_t = s_1 \dots s_n$ в T **матрицей распознавания** R для S_t в $L(G)$ назовем трехмерную двоичную матрицу $R = [r(i, j, k)]_{nnq}$ ²⁾ (где n — длина S_t и q — число символов в N), такую, что $r(i, j, k) = 1$ тогда и только тогда, когда подцепочка S_t длины i , первый символ которой есть s_j , порождается G из k -го символа $A_k \in N$, т. е. $r(i, j, k) = 1$ в том и только в том случае, когда $A_k \Rightarrow s_j s_{j+1} \dots s_{j+i-1}$.

Согласно этому определению, $r(n, 1, 1) = 1$ тогда и только тогда, когда $A_1 \Rightarrow s_1 s_2 \dots s_n$, т. е. когда S_t входит в $L(G)$. **Критерием распознавания** принадлежности S_t к $L(G)$, следовательно, является равенство $r(n, 1, 1) = 1$.

Для иллюстрации вернемся к нормальной грамматике, определенной правилами (2). Возьмем в качестве проверяемой цепочки $S_t = s_0 s_1 0 s$. Матрица распознавания для S_t в $L(G)$ выглядит так (j — номер первого символа подцепочки):

Длина i		1	2	3	4	5	6
		$S_t = s_0 s_1 0 s$					
Номер j		123456	12345	1234	123	12	1
Символ A_k	$S (= A_1)$	1.1..1		1...	..1	..	1
	$A (= A_2)$		.1..1	...1	...	.1	.
	$B (= A_3)$	.1.11.	.1.11	.1.1	.1.	.1	.

(3)

¹⁾ Этот словарь, вообще говоря, шире словаря контекстно-свободной грамматики, из которой получена данная нормальная грамматика. Добавились символы двух типов: те, которые позволяют заменить правило $A \rightarrow BC$ правилами $A \rightarrow Bc$ и $C \rightarrow c$, и те, с помощью которых правило $A \rightarrow \psi$, где ψ имеет длину $l > 2$, заменяется $l - 1$ двойным правилом.

²⁾ Имеется в виду, что i и j изменяются от 1 до n , а k — от 1 до q .

В этой матрице распознавания индексы i, j, k имеют следующие значения:

i — длина подцепочки;

j — номер первого символа подцепочки в проверяемой цепочке;

k — индекс над множеством нетерминальных символов нормальной грамматики.

Каждая подцепочка проверяемой цепочки S_i определяется двумя индексами i, j . Поскольку пары индексов, для которых $i + j > n + 1$, не соответствуют никаким подцепочкам, то соответствующие элементы матрицы опущены. Другие нулевые элементы для большей ясности обозначены точками. Кроме того, для удобства представления отдельно показаны плоскости матрицы для различных i — длин подцепочек. Аналогичное разбиение можно произвести и по другим координатам. Смысл элементов этой матрицы R таков: пусть $r(4, 2, 3) = 1$. Это означает, что подцепочка S_i длины 4, начинающаяся со второго символа, порождается грамматикой из $A_3 = B$. Для $S_i = s0s10s$ такая подцепочка имеет вид $0s10$; она действительно порождается B таким образом: $B \Rightarrow BB \Rightarrow BSB \Rightarrow BSBB \Rightarrow 0s10$.

4. АЛГОРИТМ РАСПОЗНАВАНИЯ КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ

Алгоритм распознавания есть итерационная процедура для вычисления элементов R . Для $i = 1$ элементы матрицы вычисляются, исходя из терминальных правил G , в частности для $i = 1$ $r(1, j, k) = 1$, если $A_k \Rightarrow s_j$, т. е. если $A_k \rightarrow s_j$ есть терминальное правило G , и $r(1, j, k) = 0$ в противном случае. Отыскав с помощью терминальных правил плоскость матрицы для $i = 1$, мы найдем остальные элементы матрицы, используя двойные правила. Например, $(2, 2, 2)$ -элемент матрицы распознавания оказывается равным 1 по следующим причинам. Нетерминальный символ с индексом 2 есть A [имеется в виду грамматика (2). — *Перев.*]. Следовательно, первый шаг в порождении из A должен использовать двойное правило, в котором A стоит слева. В нашей грамматике одно такое правило $A \rightarrow BS$. Поскольку A должен порождать терминальную цепочку длины 2, то B и S должны порождать цепочки длины 1 каждый. Чтобы узнать, порождает ли $B (= A_3)$ такую цепочку, рассмотрим элемент матрицы $(1, 2, 3)$; чтобы проверить $S (= A_1)$, берем $(1, 3, 1)$ -элемент. Поскольку оба этих элемента равны 1, $r(2, 2, 2) = 1$.

В общем случае получаем следующую итеративную процедуру для вычисления $R = [r(i, j, k)]_{n \times n \times q}$:

а) $r(1, j, k) = 1$, если $A_k \rightarrow s_j$ — терминальное правило G , и $r(1, j, k) = 0$ в противном случае;

б) если вычислены элементы матрицы для всех $i' < i$, где $i > 1$, то

$$r(i, j, k) = \sum_{(k_1, k_2) \in P_k} \sum_{t=1}^{i-1} r(t, j, k_1) r(i-t, j+t, k_2) \quad (\text{суммирование логическое}), \quad (4)$$

где P_k — множество упорядоченных пар (k_1, k_2) , таких, что $A_k \rightarrow A_{k_1} A_{k_2}$ — правило G . Другими словами, эта логическая сумма означает, что $r(i, j, k) = 1$, если для некоторого правила $A_k \rightarrow A_{k_1} A_{k_2}$ из G и некоторого t между 1 и $i-1$ предварительно вычисленные значения $r(t, j, k_1)$ и $r(i-t, j+t, k_2)$ оба равны 1; $r(i, j, k) = 0$ в противном случае.

Доказательство правильности итеративного процесса. Предположим, что процедура вычисляет элементы матрицы распознавания в соответствии с определением для всех $i < i_0$. Чтобы определение удовлетворялось при $i = i_0$, $r(i_0, j, k)$ должно быть равно 1 только в случае, когда $A_k \Rightarrow s_j s_{j+1} \dots s_{j+i_0-1}$. Это заведомо верно при $i_0 = 1$; для $i_0 > 1$ это выполняется в том и только в том случае, когда существует такое значение индекса t между 1 и $i_0 - 1$, что для некоторого правила $A_k \rightarrow A_{k_1} A_{k_2}$ нормальной грамматики выполняется $A_{k_1} \Rightarrow s_j s_{j+1} \dots s_{j+t-1}$ и $A_{k_2} \Rightarrow s_{j+t} s_{j+t+1} \dots s_{j+i_0-1}$. Последняя пара отношений может выполняться тогда и только тогда, когда $r(t, j, k_1) = 1$ и $r(i_0 - t, j + t, k_2) = 1$ по предположению индукции. Другими словами, $r(i_0, j, k) = 1$ тогда и только тогда, когда логическая сумма (4) для $i = i_0$ равна 1.

Пока мы вычисляем матрицу распознавания вручную, нас вполне устраивает запись правил грамматики в виде, аналогичном (2); при автоматическом вычислении требуется форма, в большей степени соответствующая этому случаю. По аналогии с матрицей распознавания представим грамматику в форме двоичной матрицы. Здесь нужны, однако, две матрицы: одна — для терминальных правил, а другая — для двойных. Терминальные правила представляются упорядоченной парой символов каждое: первый — символ нетерминальный, второй — терминальный. Правила можно представить в виде двоичной матрицы

$$G_t = [g_t(k, l)]_{qs},$$

где q — число нетерминальных символов, а s — число терминальных символов в словаре. Элемент $g_t(k, l) = 1$, если $A_k \rightarrow a_l$ — терминальное правило G , и $g_t(k, l) = 0$ в противном случае. Двойные правила суть упорядоченные тройки символов из N . Их можно представить в виде матрицы $G_c = [g_c(k, k_1, k_2)]_{qqq}$. Элементы

$g_c(k, k_1, k_2) = 1$, если $A_k \rightarrow A_{k_1} A_{k_2}$ — правило G , и в противном случае $g_c(k, k_1, k_2) = 0$.

Например, вспомним грамматику (2); терминальные правила можно представить в виде

$$G_t = \begin{array}{c|ccc} & \begin{array}{c} l \\ k \end{array} & 0 & 1 & s \\ \hline S & & . & . & 1 \\ A & & . & . & . \\ B & & 1 & 1 & . \end{array} \quad (5)$$

Аналогично двойные правила представляются двоичной матрицей

$$G_c = \begin{array}{c|ccc} & k & S & A & B \\ \hline k_2 & SAB & SAB & SAB \\ \hline k_1 \begin{array}{c} S \\ A \\ B \end{array} & \begin{array}{c|cc} . & 1 & . \\ \hline . & . & . \\ . & . & . \end{array} & \begin{array}{c|cc} . & . & . \\ \hline . & . & . \\ . & 1 & . \end{array} & \begin{array}{c|cc} . & . & . \\ \hline . & . & . \\ . & 1 & 1 \end{array} \end{array} \quad (6)$$

Применяя матричную форму грамматики, можно переписать уравнение (4), используемое для итеративного вычисления элементов R , в виде

$$r(i, j, k) = \sum_{t=1}^{i-1} \sum_{k_1=1}^q \sum_{k_2=1}^q g_c(k, k_1, k_2) r(t, j, k_1) r(i-t, j+t, k_2),$$

$$i > 1 \text{ (суммирование логическое)}. \quad (7)$$

Теперь мы в состоянии оценить число элементарных операций, требуемых для вычисления R . Пусть $T(R)$ — число логических произведений, требуемых для вычисления всех элементов R ; $T(r(i, j, k))$ означает аналогичную величину для одного элемента $r(i, j, k)$. Для простоты опустим операции вычисления для плоскости $i = 1$. Получаем из (7)

$$T(r(i, j, k)) = (i-1) q^2, \quad (8)$$

$$\begin{aligned} T(R) &= \sum_{\substack{i \geq 2, j \geq 1 \\ i+j \leq n+1}}^n \sum_{k=1}^q T(r(i, j, k)) = \sum_{\substack{i \geq 2, j \geq 1 \\ i+j \leq n+1}}^n (i-1) q^3 = \\ &= \frac{1}{6} q^3 n (n^2 - 1) < \frac{1}{6} q^3 n^3. \end{aligned} \quad (9)$$

Поскольку вычисление всех элементов матрицы распознавания может быть выполнено нахождением $(1/6)q^3 n^3$ логических

произведений, отсюда следует, что распознавание произвольного контекстно-свободного языка можно выполнить, вычислив $\frac{1}{6} q^3 n^3$ логических произведений¹⁾).

В результате получаем следующий **алгоритм распознавания**. найти с помощью итеративного процесса все элементы матрицы распознавания для S_t из $L(G)$. Если $(n, 1, 1)$ -элемент этой матрицы равен 1, то цепочка является предложением $L(G)$, а в противном случае не является таковой.

5. НЕОДНОЗНАЧНОСТЬ ПРЕДЛОЖЕНИЯ

Небольшое расширение метода распознавания позволяет использовать его для нахождения неоднозначности предложения в любом контекстно-свободном языке. **Неоднозначность** предложения относительно грамматики G есть число различных способов, которыми это предложение получается из S с помощью G . Говорят, что предложение **однозначно**, если его неоднозначность равна 1, и **неоднозначно**, если его неоднозначность больше 1; неоднозначность вычисляется тем же способом, который мы использовали при распознавании, только вместо логических операций используются обычные арифметические.

Определение. Для данной грамматики G в словаре TUN и данной цепочки S_t в T **матрица неоднозначности** для S_t из $L(G)$ есть трехмерная матрица $M = [m(i, j, k)]_{nng}$ из натуральных чисел (n — длина S_t , q — число символов в N), такая, что $m(i, j, k)$ равно числу различных способов, которыми подцепочка цепочки S_t , имеющая длину i и начинающаяся j -м символом S_t , порождается k -м символом $A_k \in N$ грамматикой G .

Из этого определения следует, что неоднозначность цепочки S_t в $L(G)$ равна $m(n, 1, 1)$. Если $m(n, 1, 1) = 0$, то S_t не содержится в $L(G)$.

Итеративная процедура для вычисления $M = [m(i, j, k)]_{nng}$:

а) $m(1, j, k) = 1$, если $A_k \rightarrow s_j$ — терминальное правило G , и $m(1, j, k) = 0$ в противном случае;

¹⁾ В порядке частного сообщения А. McKellar отметил, что общий алгоритм из [3] для оптимального кодирования алфавита из n символов двоичными словами переменной длины требует $\frac{1}{6} n(n^2 - 1)$ операций — величина, наводящая на мысль о параллелизме между этим алгоритмом и алгоритмом данной статьи. Действительно, хотя объекты этих алгоритмов совершенно различны, оба они используют подход «от конца к началу» (bottom up), что и приводит к сходству вычислительных сложностей.

б) если вычислены элементы матрицы для всех $i' \leq i$, где $i > 1$, то

$$m(i, j, k) = \sum_{(k_1, k_2) \in P_k} \sum_{t=1}^{i-1} m(t, j, k_1) m(i-t, j+t, k_2) \quad (\text{обычное арифметическое суммирование}). \quad (10)$$

Здесь P_k — множество упорядоченных пар (k_1, k_2) , таких, что $A_k \rightarrow A_{k_1} A_{k_2}$ — правило G .

Доказательство правильности этого алгоритма такое же, как и в случае матрицы распознавания, только логические операции заменяются арифметическими. Для иллюстрации снова обратимся к грамматике (2) и предложению $S_t = s0s10s$. Матрица неоднозначности для S_t из $L(G)$ имеет следующий вид:

Длина	i	1	2	3	4	5	6	(11)	
		$S_t = s0s10s$							
Номер	j	123456	12345	1234	123	12	1		
Символ	A_k	$S (= A_1)$	1.1..1		1...	..1	..		4
		$A (= A_2)$		.1..1	...1	...	.3		.
		$B (= A_3)$	.1.11.	.1.11	.1.1	.2.	.5	.	

Неоднозначность S_t из $L(G)$ равна 4, значению $m(n, 1, 1)$.

6. МАТРИЦА АНАЛИЗА

Анализ предложения в языке $L(G)$ есть описание того, как предложение порождается G ; это описание, вообще говоря, дается в виде дерева порождения. Для неоднозначного предложения существует несколько таких деревьев. Действительно, имеются такие грамматики G , что для некоторых предложений из языков, порожденных G , число различных деревьев порождения есть экспоненциальная функция длины предложения. Например, таковы все предложения языка, порожденного грамматикой $S \rightarrow SS, S \rightarrow s$.

Даже если можно было бы построить все деревья порождения для предложений с высокой неоднозначностью, это множество не дало бы много информации для «аналитика». Хотелось бы иметь поэтому более компактную форму описания грамматической структуры; матрица распознавания кажется подходящей для этой цели. Однако у этой матрицы есть недостаток: она

содержит лишнюю информацию. В частности, в ней есть ненулевые элементы, соответствующие случаям порождения данной подцепочки некоторым нетерминальным символом, в то время как при порождении всего предложения данная подцепочка этим символом не порождается. К счастью, эти лишние ненулевые элементы легко устранить, как видно из следующего определения.

Определение. Пусть даны нормальная грамматика G над словарем $N \cup T$ и цепочка $S_t = s_1 s_2 \dots s_n$ в T ; **матрицей анализа** P для S_t из $L(G)$ назовем трехмерную двоичную матрицу $P = [p(i, j, k)]_{nnq}$ (где n — длина S_t , q — число символов в N), такую, что $p(i, j, k) = 1$ в том и только в том случае, если для нетерминального символа A_k :

- 1) $A_k \Rightarrow s_j s_{j+1} \dots s_{j+i-1}$;
- 2) либо $S (= A_1) \Rightarrow s_1 s_2 \dots s_{j-1} A_k s_{j+1} \dots s_n$, либо $i, j, k = n, 1, 1$.

Для примера снова рассмотрим грамматику (2) и цепочку $s_0 s_1 s_0 s$. Матрица анализа для нее такова:

Длина i		1	2	3	4	5	6	(12)	
		$S_t = s_0 s_1 s_0 s$							
Номер j		123456	12345	1234	123	12	1		
Символ A_k	$S (= A_1)$	1.1..1		1...	..1	..	1		
	$A (= A_2)$		.1..1		...	.1	.		
	$B (= A_3)$	.1.11.	.1.1.	.1..	.1.	..	.		

Элементы матрицы анализа можно находить по данной матрице распознавания с помощью следующей процедуры.

Итеративная процедура для вычисления $P = [p(i, j, k)]_{nnq}$ из $R = [r(i, j, k)]_{nnq}$:

- а) $p(n, 1, 1) = r(n, 1, 1)$;
- б) если $p(i, j, k) = 1$, то для правил G вида $A_k \rightarrow A_{k_1} A_{k_2}$, таких, что $r(t, j, k_1) r(i-t, j+t, k_2) = 1$ для некоторого t , $1 \leq t \leq i-1$, полагаем все $p(t, j, k_1)$ и $p(i-t, j+t, k_2)$ равными единице; все остальные элементы, не равные 1 в силу этого правила, равны 0.

Доказательство. Предположим, что процедура дает значения элементов матрицы в соответствии с определением для всех $i > i_0$; мы хотим показать, что в этом случае определение удовлетворяется и при $i = i_0$. Когда $i_0 = n$, это очевидно. Для

$i_0 < n$ предположим, что процедура дает нам $p(i_0, j_0, k_0) = 1$. Тогда должно существовать такое правило $A_k \rightarrow A_{k_1} A_{k_2}$ (где либо $k_1 = k_0$, либо $k_2 = k_0$) и ненулевой элемент $p(i, j, k)$ для $i > i_0$, из которых получается значение 1 для $p(i_0, j_0, k_0)$. По предположению индукции элемент $p(i, j, k)$ вычислен в соответствии с определением, т. е. $S \Rightarrow s_1 \dots s_{j-1} A_k s_{j+i} \dots s_n$. Следовательно, $S \Rightarrow s_1 \dots s_{j-1} A_{k_1} A_{k_2} s_{j+i} \dots s_n$, откуда видно, что $p(i_0, j_0, k_0) = 1$ в соответствии с определением.

Проводя рассуждение в обратном порядке, видим, что обратное утверждение также верно; этим доказана правильность нашей процедуры.

7. МАШИНА ТЬЮРИНГА ДЛЯ РАСПОЗНАВАНИЯ И АНАЛИЗА

Выше показано, что все элементы матрицы распознавания можно найти примерно за $\frac{1}{6} q^3 n^3$ логических произведений. При доказательстве этого утверждения мы пренебрегли вычислением элементов плоскости $i = 1$ и не касались вопросов сложности нахождения и группировки сомножителей в данном логическом произведении. Чтобы точнее определить сложность вычисления матриц распознавания и анализа, мы промоделируем эти вычисления на машине Тьюринга.

Мы займемся этим в данном разделе; следующий раздел посвящен построению универсальной машины для распознавания и анализа контекстно-свободных языков, т. е. машины Тьюринга, которая проверяет любую цепочку относительно любой нормальной грамматики, поданной на вход.

Возможность такого моделирования сама по себе не удивительна, однако интересно, что вычисление матрицы распознавания или анализа может быть выполнено (если пренебрегать вычислением плоскости $i = 1$) за число шагов, которое отличается от соответствующего числа логических произведений только постоянным множителем. В работе [5] показано, что за счет расширения рабочего алфавита машины и соответствующей перекодировки вычислений постоянный множитель при вычислительной сложности может исчезнуть. Следовательно, может быть построена машина Тьюринга, которая вычисляет элементы матриц распознавания и анализа за число шагов, равное соответствующему числу логических произведений.

Мы будем работать с такой формой машины, которая удобна для исследований вычислительной сложности, начатых в [5] и продолженных в [7, 12]. В работе [5] вычисления классифицируются в зависимости от сложности выполнения их на машине Тьюринга. Эта сложность измеряется в терминах времени (т. е.

числа шагов машины), которое необходимо для вычислений, или в терминах зоны (т. е. числа квадратов ленты), необходимой для вычислений. Мы будем работать с критерием времени, но используя при этом модель, построенную в связи с изучением зоны.

Машина Тьюринга состоит из *управляющего устройства* с конечным числом состояний и конечного числа лент (рис. 1).

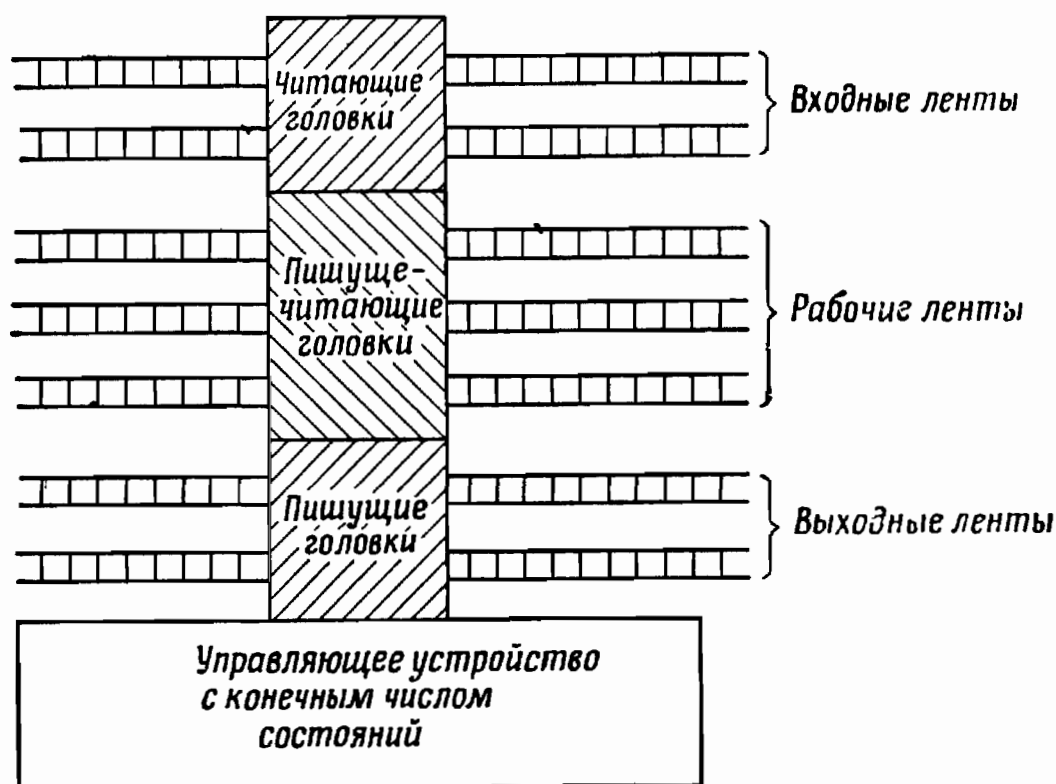


Рис. 1. Модель машины Тьюринга.

Каждая лента разделена на **квадраты**, одни из которых пусты, а другие содержат символы из данного **конечного словаря лент** (имеется в виду алфавит машины. — *Перев.*), в данном случае состоящего из двух символов 0 и 1. Связь управляющего устройства с лентами осуществляется посредством пишуще-читающих головок, по одной на каждой ленте. Каждая головка занимает один квадрат ленты. Шаг машины Тьюринга состоит из следующих трех операций:

1. Каждая головка читает (различает) символ, содержащийся в том квадрате, который она занимает¹⁾; в зависимости от этих символов управляющее устройство изменяет свое состояние.
2. Каждая головка может изменить символ, содержащийся в занимаемом ею квадрате, или не менять его.
3. Каждая головка может независимо от других передвинуть просматриваемый квадрат на единицу влево или вправо.

¹⁾ Как мы увидим, исключение делается для головок, стоящих на выходных лентах: они не могут читать символы.

Различные ленты машины играют неодинаковую роль; в зависимости от этого выделяют три класса лент:

1) Входные ленты — их можно читать, но на них нельзя писать. Поэтому каждая такая лента содержит некоторую неизменную запись. В начале работы головка, связанная с входной лентой, стоит над крайним левым непустым символом.

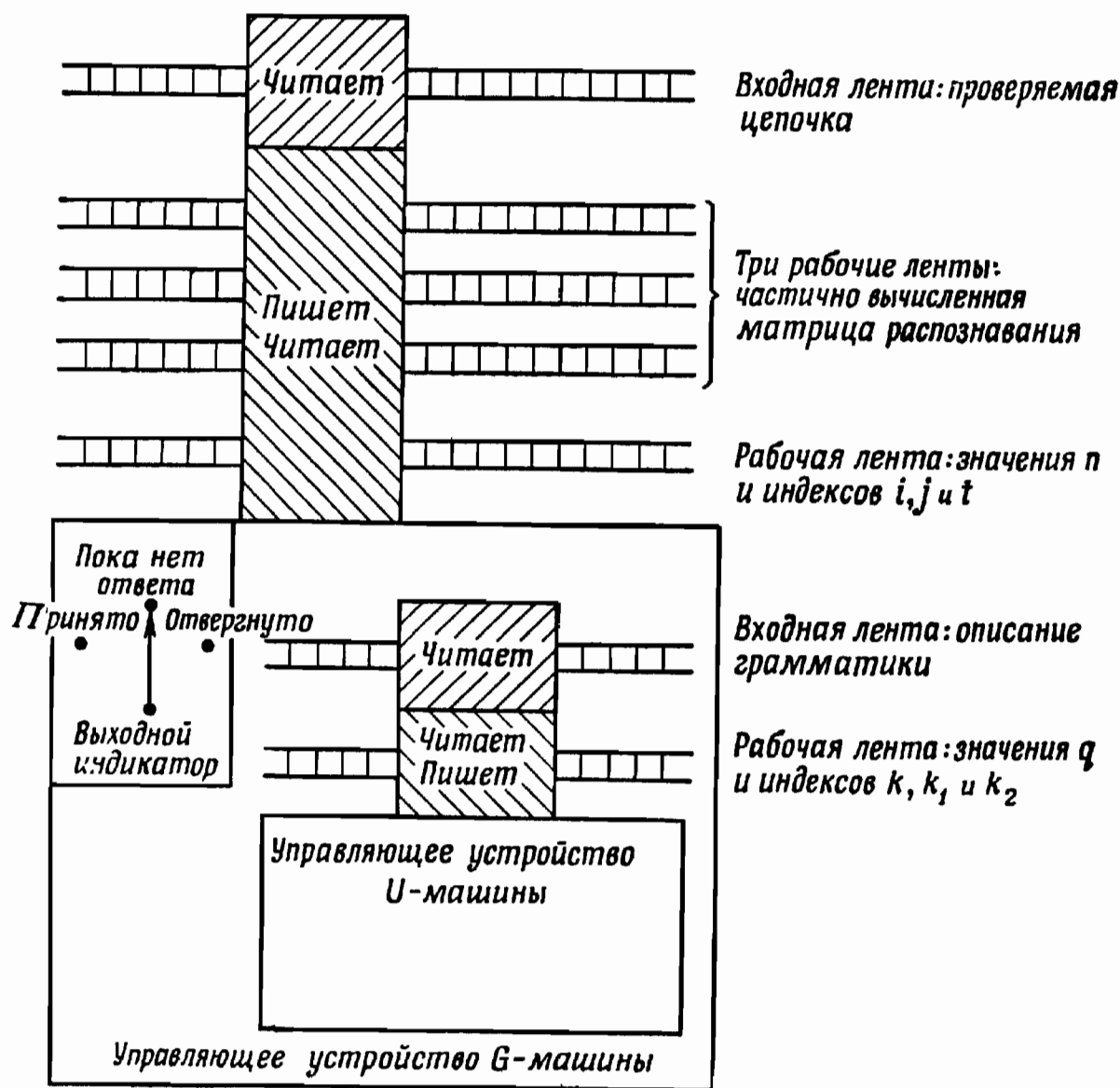


Рис. 2. Структура машины Тьюринга, реализующей алгоритм распознавания.

2) Рабочие ленты — на них можно и читать и писать. Вначале все квадраты рабочей ленты пусты.

3) Выходные ленты — на них можно писать, но их нельзя читать. Вначале все квадраты выходных лент пусты.

Теперь опишем машину Тьюринга, реализующую алгоритм распознавания. Мы будем предполагать, что правила грамматики хранятся в памяти управляющего устройства, так что наша машина распознает лишь язык конкретной грамматики G . Такую машину назовем G -машиной (рис. 2). В следующем разделе мы расчленим управляющее устройство G -машины на управ-

ляющее устройство универсальной машины (U -машины), входную ленту, на которой задана грамматика, и рабочую ленту, на которой временно хранятся текущие параметры грамматики.

G -машина имеет одну входную ленту. На этой ленте записана входная цепочка. Терминальные символы этой цепочки закодированы символами словаря лент. Для простоты предположим, что это кодирование в унарной системе, т. е. терминальный символ a_i представляется цепочкой 1 последовательных единиц, за которыми следует пустой квадрат. Головка на входной ленте стоит в начальный момент на крайнем левом непустом квадрате. Эта головка может сдвигаться только вправо, воспринимая одиночный пустой квадрат как конец символа, а пару пустых квадратов как конец цепочки.

G -машина имеет четыре рабочие ленты. На трех из них расположены элементы матрицы распознавания, развернутой линейно¹⁾. На четвертой ленте записаны значения n и индексов i , j и t .

G -машина не нуждается в выходной ленте. Результат распознавания отражается на специальном индикаторе, стрелка которого занимает одно из трех положений: «принято» (т. е. данная цепочка входит в $L(G)$), «отвергнуто» и «пока нет ответа».

Объясним подробнее, в каком виде появляются на лентах элементы матрицы распознавания. Напомним, что элементы матрицы распознавания имеют значения 0 или 1. Просматриваемая часть лент, на которых вычисляется матрица распознавания, не содержит пустых символов: в каждом квадрате записан 0 или 1. Элемент $r(i, j, k)$ матрицы находится в $[(i-1)nq + (j-1)q + k]$ -м квадрате каждой из трех лент; отсчет квадратов ведется слева направо от начального положения головки; квадрату, на котором стоит головка перед распознаванием, присваивается номер 1. Согласно этой формуле, элементы, соседние по индексу i (и с совпадающими остальными индексами), разделены на ленте $nq - 1$ квадратами, соседние по индексу j разделены $q - 1$ квадратами и соседние по индексу k оказываются соседними и на ленте.

Теперь опишем работу G -машины при распознавании. Она строится согласно итеративной процедуре вычисления R , данной в разд. 4. В первую очередь вычисляются элементы матрицы распознавания для $i = 1$: проверяемая цепочка читается слева направо и после прочтения каждого символа цепочки управляющее устройство с помощью внутренней памяти находит

¹⁾ Вместо трех лент можно было бы использовать одну ленту с тремя головками. Такая модель была бы обобщением рассматриваемой здесь.

все нетерминальные символы, которые порождают данный терминальный посредством применения единственного терминального правила. Если оказывается, что j -й символ цепочки порождается нетерминальным символом A_k , то $r(i, j, k) = 1$. Все остальные элементы для $i = 1$ полагаются равными 0. После получения всех элементов для $i = 1$ на одной из рабочих лент они переписываются на две остальные ленты, предназначенные для вычисления матрицы R .

Вычисляемая вслед за этим плоскость $i = 2$ сначала также получается на одной ленте, а затем переносится на остальные. Используя каждый раз уравнение (4), мы продолжаем вычисления для все больших значений i . В силу (4) каждый элемент матрицы распознавания определяется парами вычисленных ранее элементов, причем выбор самих пар определяется двойными правилами грамматики. Две рабочие ленты используются как память для определения значений этих пар уже вычисленных элементов, а на третьей записывается результат, вычисленный в зависимости от конъюнкций единиц среди выбранных пар.

Элементы матрицы R с индексом $i + 1$ вычисляются только после того, как вычислены все элементы с индексом i . При данном i элемент матрицы требует для вычисления $i - 1$ проход; каждый проход соответствует одному значению t в уравнении (4). Во время каждого прохода элементы матрицы вычисляются в порядке возрастания j . Кроме того, для данных i, t, j вычисление происходит в порядке возрастания k ; в этом случае головка, записывающая результат, двигается только слева направо. Для сдвига этой головки на один квадрат головки, собирающие информацию, должны пересмотреть все возможные пары индексов k_1, k_2 . Так работает машина Тьюринга, реализующая алгоритм распознавания.

G -машину легко превратить в G -анализатор. Достаточно добавить выходную ленту, на которой будет записываться матрица анализа (точно в таком же виде, как и матрица распознавания). Сначала G -анализатор вычисляет описанным выше образом матрицу распознавания. Если $r(n, 1, 1) = 0$, то проверяемая цепочка не есть предложение из $L(G)$, и матрица анализа полностью нулевая. Если же $r(n, 1, 1) = 1$, то ненулевые элементы матрицы распознавания, не связанные с грамматической структурой, обращаются в 0 в соответствии с итеративным соотношением разд. 4. Это по сути дела та же процедура, которая используется для вычисления матрицы распознавания, но идет она «от конца к началу» — от значения n в порядке убывания i . Очевидно, число шагов здесь не больше, чем в случае вычисления матрицы распознавания.

8. УНИВЕРСАЛЬНАЯ МАШИНА ДЛЯ РАСПОЗНАВАНИЯ И АНАЛИЗА КОНТЕКСТНО-СВОБОДНЫХ ЯЗЫКОВ

Теперь расчленим управляющее устройство G -машины. Оно состоит из управляющего устройства U -машины (U означает универсальность) и двух лент. Другими словами, добавив две ленты для работы с грамматикой и изменив управляющее устройство, мы можем создать U -машину, которая будет выполнять распознавание для **любой** контекстно-свободной грамматики в нормальной форме. Одна из дополнительных лент — входная; на ней записана грамматика в виде трех массивов, разделенных одиночными пустыми квадратами. Первый массив — цепочка единиц длины q — число нетерминальных символов нормальной грамматики. Второй массив — это терминальные правила грамматики. Он занимает qs квадратов (s — число терминальных символов). Квадрат, отстоящий на $(l-1)q + k$ квадратов от ближайшего левого пустого квадрата, содержит 1, тогда и только тогда, когда $A_k \rightarrow a_l$ есть терминальное правило, и 0 в противном случае. Третий массив занят двойными правилами грамматики и занимает q^3 квадратов. Здесь квадрат, отстоящий на $(k-1)q^2 + (k_1-1)q + k_2$ квадратов от ближайшего левого пустого квадрата, содержит 1, если $A_k \rightarrow A_{k_1}A_{k_2}$ — двойное правило грамматики, и 0 в противном случае. Вид входной ленты выбран в соответствии с формой записи матрицы распознавания на ленте.

Вторая добавочная лента — это рабочая лента, на которой хранятся записи параметров грамматики, т. е. q, k, k_1, k_2 .

Теперь мы можем сосчитать число шагов, необходимое для распознавания U -машине. Оно оказывается похожим на ранее полученную величину, именно $\frac{1}{6}q^3n^3$. Теперь мы, однако, должны учесть число шагов, требуемое для вычисления плоскости $i=1$, для возвращения рабочих и читающих головок в исходные положения, для записи двух дополнительных копий матрицы распознавания. Все это увеличивает время распознавания. Не вдаваясь в подробности, скажем только, что для вычисления плоскости $i=1$ нужно не более чем $(2s+1)qn$ шагов, для остальных элементов — не более чем $\frac{1}{3}n^3(q^3+1) + 2qn^2$. Вместе это дает для времени распознавания оценку сверху

$$\frac{1}{3}n^3(q^3+1) + (2s+2n+1)qn.$$

При $q > 1, qn \gg s$ получаем

$$T_r = \frac{1}{2}n^3q^3, \quad (13)$$

где T_r — обозначает верхнюю границу времени распознавания.

Для U -анализатора время вычислений превосходит время распознавания не более чем вдвое, поэтому при $qn \gg s$, $q > 1$

$$T_p = n^3 q^3. \quad (14)$$

9. ЗАМЕЧАНИЯ

Алгоритмы для распознавания и анализа, требующие числа шагов, пропорционального n^3 , приводят к вопросу о существовании контекстно-свободных языков, которые в самом деле требуют такого большого числа шагов. Получить такой результат оказывается трудно. Действительно, в то время как в [5] для некоторого конкретного контекстно-свободного языка доказано, что его распознавание требует более чем n шагов, до сих пор не показано, что существуют контекстно-свободные языки, требующие для распознавания время, большее, чем любая пропорциональная n величина, хотя интуитивно чувствуется, что такие языки существуют¹⁾.

Если существует язык, требующий для распознавания αn^3 операций (α — некоторая константа), то он не порождается металинейной грамматикой, потому что с помощью матричного метода можно показать, что все такие языки распознаются за βn^2 операций (β — некоторая константа)²⁾.

ЛИТЕРАТУРА

1. Chomsky N., Formal properties of grammars, In «Handbook of Mathematical Psychology», v. II, Wiley, New York, 1963. (Русский перевод: Кибернетический сборник, новая серия, вып. 2, 1966, 121—230.)
2. Chomsky N., On certain formal properties of grammars, *Information and Control*, 2 (1959), 137—167. (Русский перевод: Кибернетический сборник, вып. 5, 1962, 279—311.)
3. Gilbert E. N., Moore E. F., Variable-length binary encodings, *Bell System Techn. J.*, 38 (1959), 933—967. (Русский перевод: Кибернетический сборник, вып. 3, 1961, 103—141.)
4. Griffiths T. V., Petrick S. R., On the relative efficiencies of context-free grammar recognizers, *Commun. Assoc. Comput. Mach.*, 8 (1965), 283—300.
5. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, 117 (1965), 285—306. (Русский перевод: Кибернетический сборник, новая серия, вып. 4, 1967, 57—85.)

¹⁾ По этому поводу см. статью Ф. Хенни в настоящем сборнике, стр. 249—270. — *Прим. ред.*

²⁾ Следует отметить недавно полученные А. Б. Галачянцем результаты: для всякого контекстно-свободного языка существует двумерная итеративная сеть, распознающая его за время C_n ; существует трехмерная итеративная сеть, универсальная для контекстно-свободных языков, распознающая их за время cqn . — *Прим. ред.*

- 6 Hays D., Automatic language-data processing. In «Computer Applications in the Behavioral Sciences», Prentice Hall, Englewood Cliffs, New Jersey, 1962.
7. Hennie F. C., Stearns R. E., Two-tape simulation of multitape Turing machines, Gen. Elec. Res. Lab. Report № 65-RL-4020E, 1965¹⁾.
8. Kay M., A parsing procedure, In «Information Processing 1962», North Holland, Amsterdam, 1963.
9. Kuno S., Oettinger A. E., Multiple-path syntactic analyser. In «Information Processing 1962», North Holland, Amsterdam, 1963.
10. Kuno S., The predictive analyser and a path elimination technique, *Commun. Assoc. Comput. Mach.*, 8 (1965), 453—463.
11. Kasami T., An efficient recognition and syntax-analysis algorithm for context-free languages, University of Illinois Coordinated Science Lab. Rept. № R-257, 1966.
12. Lewis P. M. II, Stearns R. E., Hartmanis J., Memory bounds for recognition of context-free and context-sensitive languages, *IEEE Conf. Record Switching Circ. Theory and Log. Des.*, 1965, 191—202. (Настоящий сборник, стр. 320—338.)
13. Sakai I., Syntax in universal translation, *Proc. 1961 Intern. Conf. Mach. Transl. Languages and Appl. Language Analysis* (Her Majesty's Stationery Office, London), 1962.
14. Stearns R. E., Hartmanis J., Lewis P. M. II, Hierarchies of memory limited computations, *IEEE Conf. Record Switching Circ. Theory and Log. Des.*, 1965, 179—190. (Настоящий сборник, стр. 301—319.)

¹⁾ См. также одноименную статью тех же авторов на стр. 194—212 настоящего сборника. — *Прим. ред.*

О ВРЕМЕНИ МАШИННОГО РАСПОЗНАВАНИЯ ЯЗЫКОВ, ПОРОЖДЕННЫХ ЛИНЕЙНЫМИ ГРАММАТИКАМИ ¹⁾

Т. Касами

Обозначим через V_T конечное множество символов и через V_T^* множество конечных последовательностей (в том числе и пустую последовательность) элементов из V_T . Будет использоваться модель машины Тьюринга со входом ²⁾, такая же, как в [2, 3]. Машина Тьюринга, помимо рабочих лент, имеет одностороннюю входную ленту с символами из V_T и одностороннюю выходную с символами 1 и 0. Говорят, что машина Тьюринга распознает подмножество $L \subseteq V_T^*$ в том и только в том случае, когда для любой входной последовательности длины n в V_T n -й выходной символ есть 1, если эта последовательность содержится в L , и 0 в противном случае. Говорят также, что L $T(n)$ -распознаваемо, если существует машина Тьюринга, которая распознает L , причем для любой входной последовательности печатает n -й выходной символ не более чем за $T(n)$ операций. Подробности можно найти в работах [2, 3], определения контекстно-свободных грамматик и языков, а также линейных грамматик ³⁾ — в [1].

В работе [3] построено множество A последовательностей, для которого можно найти весьма хорошую (для этого множества) нижнюю границу времени вычислений (т. е. числа операций машины) при распознавании на многоленточной машине Тьюринга. Здесь мы используем с небольшими изменениями это множество A .

Множество A' есть множество конечных последовательностей в алфавите $\{0, 1, s, c\}$, которые удовлетворяют следующим условиям:

а) первый и последний символы последовательности суть s , а сама она состоит из блоков нулей и единиц одинаковой длины, разделенных символами s или scs ;

¹⁾ Kasami T., A note on computing time for recognition of languages generated by linear grammars, *Information and Control*, 10, № 2 (1967), 209—214.

²⁾ В подлиннике on-line Turing machine; при переводе здесь и далее мы называем ее просто машиной Тьюринга. — Прим. перев

³⁾ Контекстно-свободная грамматика называется линейной, если каждое ее нетерминальное правило имеет вид $A \rightarrow xBy$, где $A, B \in V_N$, $x, y \in V_T^*$. — Прим. ред.

б) если длина блока есть k , то число блоков превышает 2^k . 2^k -й и $(2^k + 1)$ -й блоки разделены scs , а все другие последовательные блоки — символом s ;

в) зеркальное отражение последнего блока нулей и единиц совпадает с каким-нибудь блоком, содержащимся среди первых блоков.

С помощью тех же рассуждений, что и в [3], получаем лемму 1.

Лемма 1. Если $A' T(n)$ -распознаваемо на машине Тьюринга, то

$$T(n) \geq C \left(\frac{n}{\log n} \right)^2,$$

где C — константа, зависящая только от числа лент, числа символов в алфавите машины и числа ее внутренних состояний.

Пусть $R(\varphi)$ означает обращение последовательности φ , например, если $\varphi = 011$, $R(\varphi) = 110$; пусть

$$L_0 = \{\varphi_0 s \varphi_1 s \varphi_2 c \varphi_3 s R(\varphi_1) s \mid \varphi_0, \varphi_2, \varphi_3 \in \{0, 1, s\}^*, \varphi_1 \in \{0, 1\}^*\}.$$

Легко видеть, что L_0 порождается линейной грамматикой $G_0 = (V_{N_0}, V_{T_0}, P_0, S)$, где множество терминальных символов $V_{T_0} = \{0, 1, s, c\}$ множество нетерминальных символов $V_{N_0} = \{S, X, Y\}$, S — начальный символ и P_0 — множество порождающих правил:

$$\begin{array}{lll} S \rightarrow 0S, & X \rightarrow 0X0, & Y \rightarrow 0Y, \\ S \rightarrow 1S, & X \rightarrow 1X1, & Y \rightarrow 1Y, \\ S \rightarrow sS, & X \rightarrow sYs, & Y \rightarrow sY, \\ S \rightarrow sXs, & Y \rightarrow c, & Y \rightarrow Y0, \\ & & Y \rightarrow Y1, \\ & & Y \rightarrow Ys. \end{array}$$

Язык L_0 можно считать одним из вариантов языка, который не является n -распознаваемым (доказательство в [2]).

Следующая лемма вытекает непосредственно из определений множеств A' и L_0 .

Лемма 2. Если последовательность φ в алфавите $\{0, 1, s, c\}$ удовлетворяет условиям а) и б) определения множества A' , то $\varphi \in A'$ в том и только в том случае, когда $\varphi \in L_0$.

Распознать, удовлетворяет ли данная последовательность условиям а) и б), можно с помощью вычислений в реальное

время (см. [7]). Поэтому лемма 2 дает нам, что если L_0 $T(n)$ -распознаваемо на машине Тьюринга, то A' также $T(n)$ -распознаваемо. Из леммы 1 получаем теорему.

Теорема 1. Если L_0 $T(n)$ -распознаваемо на многоленточной машине Тьюринга, то существует константа C , такая, что

$$T(n) \geq C \left(\frac{n}{\log n} \right)^2.$$

С другой стороны, справедлива следующая теорема.

Теорема 2. Всякий язык, порожденный линейной грамматикой, Cn^2 -распознаваем на одноленточной машине Тьюринга (C — константа).

Эта теорема доказана в работе [4]. Более простое доказательство приведено в дополнении.

Показано, что любой контекстно-свободный язык n^3 -распознаваем на машине Тьюринга (см. [5, 8, 6]¹⁾). Между оценками $\left(\frac{n}{\log n} \right)^2$ и n^3 имеется разрыв.

ДОПОЛНЕНИЕ

Доказательство теоремы 2. Пусть $G = (V_N, V_T, P, S)$ — линейная грамматика; V_N — конечное множество нетерминальных символов, V_T — конечное множество терминальных символов, P — множество правил, S — начальный символ. Пусть $V = V_N \cup V_T$ и V^* — множество всех конечных последовательностей V (включая пустую последовательность). Мы пишем

$$\Psi_1 \rightarrow \Psi_2,$$

где $\Psi_1 = \omega_1 X \omega_2$ ($\omega_1, \omega_2 \in V^*$, $X \in V_N$), $\Psi_2 = \omega_1 \varphi \omega_2$, если $X \rightarrow \varphi$ — правило грамматики G . Далее мы пишем $\Phi \Rightarrow \Psi$, если существует последовательность $\Phi_0, \Phi_1, \dots, \Phi_n$, такая, что $\Phi_0 = \Phi$, $\Phi_n = \Psi$ и $\Phi_{j-1} \rightarrow \Phi_j$, $j = 1, \dots, n$. Теперь язык L , порожденный G , определяется как

$$L = \{ \Psi \mid S \Rightarrow \Psi, \Psi \in V_T^* \}. \quad (1)$$

Без потери общности можно предполагать, что правила грамматики G имеют вид

$$Y \rightarrow aX, \quad Y \rightarrow Xa \quad \text{или} \quad Y \rightarrow a, \quad (2)$$

¹⁾ Машины из работ [5, 6] используют две рабочие ленты, а из работы [8] — четыре.

где $a \in V_T$, $Y \in V_N$ и $X \in V_N$ (см. [1]). Пусть $a_i \in V_T$ есть i -й входной символ, и пусть

$$N(j, i) = \{Y \mid Y \Rightarrow a_j \dots a_i, Y \in V_N\} \quad (1 \leq j \leq i). \quad (3)$$

Тогда из (1) и (3) следует, что последовательность $a_1 \dots a_i$ содержится в L тогда и только тогда, когда $S \in N(1, i)$ (см. [8, 6]). Если $Y \Rightarrow a_j \dots a_i$, $j < i$, то в силу (2) существует X , такое, что либо

$$X \Rightarrow a_j \dots a_{i-1} \quad \text{и} \quad Y \rightarrow Xa_i \in P,$$

либо $X \Rightarrow a_{j+1} \dots a_i$ и $Y \rightarrow a_j X \in P$. Следовательно, для $1 \leq j < i$ имеем

$$\begin{aligned} N(j, i) &= N_1(j, i) \cup N_2(j, i), \\ N_1(j, i) &= \{Y \mid Y \rightarrow Xa_i \in P, X \in N(j, i-1)\}, \\ N_2(j, i) &= \{Y \mid Y \rightarrow a_j X \in P, X \in N(j+1, i)\}. \end{aligned} \quad (4)$$

Так, если известны все $N(j, i-1)$ ($1 \leq j \leq i-1$) и $N(i, i)$, то можно по формулам (4) найти последовательно $N(i-1, i)$, $N(i-2, i)$, ..., $N(1, i)$.

Пример. Рассмотрим несложную грамматику G_1 , для которой $V_N = \{S, X\}$, $V_T = \{0, 1, c\}$ и $P = \{S \rightarrow 0S, S \rightarrow 1X, X \rightarrow S1, S \rightarrow c\}$. Пусть $a_1 a_2 a_3 a_4 = 10c1$. Тогда имеем

$$\begin{aligned} N(1, 1) &= N(1, 2) = N(2, 2) = \emptyset, \\ N(3, 3) &= \{S\}, \\ N_1(2, 3) &= \emptyset, \quad N_2(2, 3) = \{S\}, \quad N(2, 3) = \{S\}, \\ N_1(1, 3) &= N_2(1, 3) = N(1, 3) = \emptyset, \\ N(4, 4) &= \emptyset, \\ N_1(3, 4) &= \{X\}, \quad N_2(3, 4) = \emptyset, \quad N(3, 4) = \{X\}, \\ N_1(2, 4) &= \{X\}, \quad N_2(2, 4) = \emptyset, \quad N(2, 4) = \{X\}, \\ N_1(1, 4) &= \emptyset, \quad N_2(1, 4) = \{S\}, \quad N(1, 4) = \{S\}. \end{aligned}$$

Процедура выполняется на одноленточной машине Тьюринга со входом следующим образом. Разделим полуограниченную рабочую ленту на секции равной длины $T_1, \dots, T_j, \dots$. Каждая секция T_j состоит из двух частей T_{0j} и T_{1j} , в первой из которых может храниться любой терминальный символ с отметкой *, а во второй — любое подмножество V_N . Вначале все секции пустые. Предположим, что управляющее устройство (с конечным числом состояний) имеет три блока памяти M_0 , M_1 и M_2 . Блок M_0 используется для запоминания последнего входного символа, M_1 и M_2 могут хранить любое подмножество V_N . Пусть $[T_{hj}]$

($h=0, 1$; $j=1, 2, \dots$) и $[M_h]$ ($h=0, 1, 2$) обозначают множества символов, хранящихся в T_{hj} и M_h соответственно. Индексы i и j будут использоваться для описания работы этой машины. Индекс i означает, что прочитан i -й входной символ (считая от начала входного слова), а индекс j — что головка на ленте находится в секции T_j и отыскивает $N(j, i)$. Вот как работает наша машина.

1. Машина читает первый входной символ a_1 на входной ленте и пишет $*a_1$ на T_{01} , запоминает все нетерминальные символы Y , такие, что $Y \rightarrow a_1$ есть правило G , в секции T_{11} . Тогда $[T_{11}] = N(1, 1)$. Если T_{11} содержит начальный символ S , машина пишет 1 на выходной ленте, в противном случае пишет 0. Сдвигает головку на секцию T_{02} .

2. Предположим, что

$$\begin{aligned} [T_{01}] &= *a_1, \\ [T_{0k}] &= a_k & (1 < k < i), \\ [T_{1k}] &= N(k, i-1) & (1 \leq k < i) \end{aligned}$$

и что головка на секции T_{0i} . Тогда машина читает i -й входной символ на входной ленте и запоминает a_i в M_0 и T_{0i} . Запоминает все нетерминальные символы Y , такие, что $Y \rightarrow a_i$ — правило G , в секции T_{1i} и в M_2 . Тогда

$$[T_{1i}] = [M_2] = N(i, i).$$

Сдвигает головку на T_{1i-1} .

2.1. Положим, что головка на секции T_{1j} ($1 \leq j < i$) и что

$$\begin{aligned} [T_{1k}] &= N(k, i-1) & (1 \leq k \leq j), \\ [T_{1k}] &= N(k, i) & (j < k \leq i), \\ [M_0] &= a_i, \\ [M_2] &= [T_{1j+1}] = N(j+1, i). \end{aligned}$$

Тогда машина запоминает в M_1 все нетерминальные символы Y , такие, что для некоторого X в T_{1j} $Y \rightarrow Xa_i$ есть правило G . По определению

$$[M_1] = N_1(j, i).$$

Машина сдвигает головку на секцию T_{0j} , где содержится a_j (или $*a_1$, если $j=1$), запоминает в M_1 все нетерминальные символы Y , такие, что для некоторого X из M_2 $Y \rightarrow a_jX$ есть правило G . Тогда по определению

$$[M_1] = N_1(j, i) \cup N_2(j, i) = N(j, i).$$

Если T_{0j} не содержит *, то машина переписывает содержимое M_1 в M_2 и T_{1j} , сдвигает головку на T_{1j-1} и повторяет ту же процедуру для $j-1$. Если T_{0j} содержит *, то она переписывает содержимое M_1 в $T_{1j} = T_{11}$. Если в M_1 находится S , то пишет 1 на выходной ленте, в противном случае пишет 0. Сдвигает головку на секцию T_{0i+1} , которая является первой пустой секцией среди секций вида T_{0k} .

Для каждого i число шагов, требуемое для получения i -го выходного символа после прочтения i -го входного, ограничено сверху величиной Ci с константой C , не зависящей от i . Следовательно, число шагов, требуемое для получения i -го выходного символа с момента начала вычислений, ограничено сверху величиной $\frac{1}{2} Ci^2$.

Обобщение теоремы 2 см. в работе [6].

ЛИТЕРАТУРА

1. Chomsky N., Formal properties of grammars, «Handbook of Mathematical Psychology», v. II, New York, 1963. (Русский перевод: Кибернетический сборник, новая серия, вып. 2, 1966.)
2. Hartmanis J., Stearns R. E., Computational complexity of recursive sequences, Proc. Fifth Ann. Symp. Switching Circuit Theory and Logical Design, 1964, 82—90.
3. Hennie F. C., On-line Turing machine computations, *IEEE Trans. Electr. Comput.*, EC-15 (1966), 35—44. (Настоящий сборник, стр. 249—270.)
4. Kasami T., Some results on capabilities of one-dimensional iterative logical networks and their related problems, Report of University of Hawaii, Contract № AF 19 (628)-4379, № 4, 1965.
5. Kasami T., An efficient recognition and syntax-analysis algorithm for context-free languages, Report of University of Hawaii, Contract № AF19 (628)-4379, № 2, 1965.
6. Torii K., Kasami T., Ozaki H., Recognition and syntax-analysis of context-free languages, Report of Institute of Electrical Communication Engineers of Japan, June 1966.
7. Yamada H., Real-time computation and recursive functions not real-time computable, *IRE Trans. on Electron. Comput.*, EC-11 (1962), 753—760. (Настоящий сборник, стр. 139—155.)
8. Younger D. H., Recognition and parsing of context-free languages in time n^3 *Inform. Control*, 10, № 2 (1967), 189—208. (Настоящий сборник, стр. 344—362.)

РАСПОЗНАВАНИЕ МЕТАЛИНЕЙНЫХ ЯЗЫКОВ ПО МЕТОДУ Т. КАСАМИ

М. В. Ломковская

Предложенный Касами способ распознавания линейных языков допускает обобщение на класс металинейных языков.

Металинейной грамматикой называется бесконтекстная грамматика $G = \langle V_T, V_N, S, R \rangle$, у которой множество правил удовлетворяет следующим условиям:

1) $R = R_{\text{лин}} \cup R_{\text{нач}}$, где $R_{\text{лин}}$ — множество линейных правил, а $R_{\text{нач}}$ — множество начальных правил вида $S \rightarrow \omega_k$ ($k = 1, \dots, r$); здесь S — начальный символ грамматики, а ω_k — непустая цепочка нетерминальных символов, не содержащая S ;

2) начальный символ S не встречается ни в левых, ни в правых частях линейных правил грамматики.

Как нетрудно видеть, металинейный язык представляет собой конечное объединение конечных соединений (конкатенаций) линейных языков. Например, если начальные правила грамматики суть $S \rightarrow A_1 A_2 A_3$, $S \rightarrow A_2 A_1$, $S \rightarrow A_1$, $S \rightarrow A_3 A_3$, то, обозначив символом L_i язык, порождаемый линейной грамматикой $G_i = \langle V_T, V_N, A_i, R_{\text{лин}} \rangle$, имеем $L(G) = L_1 L_2 L_3 \cup L_2 L_1 \cup L_1 \cup L_3 L_3$.

В дальнейшем нам потребуются следующие обозначения: обозначим через $\Omega = \{\omega_1, \dots, \omega_r\}$ множество правых частей начальных правил, а через $\Phi = \{\varphi_1, \dots, \varphi_q\}$ множество всех различных непустых собственных начал цепочек из Ω . Для нашего примера $\Omega = \{A_1 A_2 A_3, A_2 A_1, A_1, A_3 A_3\}$, $\Phi = \{A_1, A_1 A_2, A_2, A_3\}$.

Пусть теперь дана входная цепочка $a_1 \dots a_j \dots a_i \dots a_n$. Сохраним обозначение $N(j, i)$ Касами для множества нетерминальных символов, из которых можно вывести подцепочку $a_j \dots a_i$ входной цепочки, пользуясь только линейными правилами. Удобно представить запись множества $N(j, i)$ в виде последовательности разрядов, в которых стоит 0 или 1 в зависимости от того, выводима ли данная подцепочка из символа $A \in V_N$ по линейным правилам. Через $M_\Phi(j, i)$ обозначим величину, которая равна 1, если подцепочку $a_j \dots a_i$ входной цепочки можно вывести из $\varphi \in \Phi$ в данной грамматике, и равна 0 в противном случае. Например, $M_S(1, i) = 1$ тогда и только тогда, когда цепочка $a_1 \dots a_i$ принадлежит языку $L(G)$, т. е. $M_S(1, i)$ — это i -й выходной сигнал. Для построения этого выходного сигнала

будут использованы следующие формулы:

$$M_S(1, i) = \bigvee_{\omega \in \Omega} M_\omega(1, i);$$

$$M_{\Phi_A}(1, i) = \bigvee_{1 < j \leq i} [M_\Phi(1, j-1) \wedge M_A(j, i)]. \quad (1)$$

Сигнал $M_A(j, i)$ известен, если уже построено множество $N(j, i)$; в алгоритме Касами это происходит во время i -го пробега машины. Заметим, что к этому моменту уже построены все множества $N(1, j-1)$, где $1 < j \leq i$.

Предлагаемый ниже алгоритм позволяет за время i -го пробега машины найти все множества $M_\Phi(1, i)$ ($\Phi \in \Phi \cup \Omega$), а также выходной сигнал $M_S(1, i)$. Этим решается задача распознавания металинейных языков на машине Тьюринга со входом.

Переорганизуем машину, распознающую линейные языки, следующим образом:

1. В блок состояний добавляем еще одну рабочую ячейку M_3 . Эта ячейка имеет $q + 1$ разряд, где q — мощность множества Φ ; по одному разряду предназначено для хранения $\phi \in \Phi$, и один — для выходного сигнала. Во время i -го пробега машины каждый разряд M_3 действует как накопитель: вновь пришедший сигнал логически суммируется с содержимым разряда. За время i -го пробега в ячейке M_3 по формулам (1) постепенно формируются сигналы $M_\Phi(1, i)$ и $M_S(1, i)$.

2. В каждый блок ленты, содержащий входной символ a_j ($j > 1$), добавим q разрядов для хранения сигналов $M_\Phi(1, j-1)$ ($\Phi \in \Phi$). Таким образом, j -й блок на ленте ($1 < j \leq i$) после i -го пробега машины будет иметь следующий вид (первый блок заполняется как для линейного языка):

$M_{\Phi_1}(1, j-1)$	$M_{\Phi_2}(1, j-1)$	$\dots$	$M_{\Phi_q}(1, j-1)$	a_j	$N(j, i)$

Алгоритм распознавания металинейного языка работает так:

Первый шаг. Первый шаг работы машины начинается так же, как и для линейного языка, а именно машина считывает первый входной символ a_1 и строит множество $N(1, 1)$. Затем символы из $N(1, 1)$ проверяются на принадлежность к Ω и к Φ ; сигналы $M_\omega(1, 1)$ засылаются в разряд ячейки M_3 , предназначенный для выходного сигнала, а сигналы $M_\Phi(1, 1)$ — в предназначенные для них разряды M_3 . Легко видеть, что после этой операции в M_3 окажутся правильно сформированные выходной сигнал $M_S(1, 1)$ и все символы $M_\Phi(1, 1)$ (заметим, что для более чем однобуквенной цепочки ϕ всегда $M_\Phi(1, 1) = 0$, поскольку в грамматике G из длинной цепочки ϕ нельзя получить более ко-

роткую цепочку a_1 — грамматика неукорачивающая). Выходной сигнал подается на выход, а символы $M_\Phi(1, 1)$ хранятся в ячейке M_3 во время обратного движения по ленте и печатаются на ленту в начале следующего, второго блока.

i-й шаг. Пусть после $(i - 1)$ -го пробега по ленте на ней заполнен $i - 1$ блок, выдан $(i - 1)$ -й выходной символ и вычислены символы $M_\Phi(1, i - 1)$, которые печатаются на ленте в начале i -го блока. Ячейка M_3 пуста. Машина считывает очередной входной символ a_i и строит множество $N(i, i)$. После этого каждый блок, начиная с i -го и кончая вторым, машина обрабатывает следующим образом:

Обработка j-го блока.

1. Вычисляется множество $N(j, i)$ так же, как и в случае линейного языка (для i -го блока это уже сделано).

2. Для любого ϕ из Φ машина находит все $A \in V_N$, такие, что $\phi A \in \Phi \cup \Omega$. Образует последовательно логические произведения $M_\Phi(1, j - 1) \wedge M_A(j, i)$ (оба сомножителя записаны в обрабатываемом блоке, а в качестве рабочих ячеек можно использовать M_1 и M_2 , в данный момент пустые, увеличив их объем, если потребуется). Засылает каждое такое произведение в разряд ячейки M_3 , предназначенный для $M_{\phi A}(1, i)$, если $\phi A \in \Phi$, и в разряд для выходного сигнала, если $\phi A \in \Omega$. В M_3 по формулам (1) накапливаются сигналы $M_\Phi(1, i)$ и $M_S(1, i)$.

3. Переход к $(j - 1)$ -му блоку.

Обработка первого блока.

1. Вычисляется $N(1, i)$.

2. Как на первом шаге, элементы из $N(1, i)$ проверяются на принадлежность к Φ и к Ω . Символы $M_A(1, i)$ ($A \in \Phi$) заполняют оставшиеся незаполненными разряды M_3 , соответствующие однобуквенным началам; символы $M_A(1, i)$ ($A \in \Omega$) засылаются в разряд для выходного сигнала (где по-прежнему происходит логическое суммирование с уже имеющимся в разряде символом). Очевидно, после этого в ячейке M_3 образуется выходной сигнал, а также все символы $M_\Phi(1, i)$. Выходной сигнал подается на выход, а символы $M_\Phi(1, i)$ хранятся в ячейке M_3 во время обратного движения по ленте и печатаются на ленту в начале $(i + 1)$ -го блока; ячейка M_3 пуста.

Описание работы алгоритма закончено. Легко видеть, что по сравнению с алгоритмом распознавания линейных языков число шагов по параметрам i и j не меняется: удлиняется время обработки j -го блока, но это время по-прежнему можно ограничить сверху константой C , зависящей от грамматики (в частности, C зависит от мощностей алфавитов, от множества $R_{\text{лин}}$ и от множества Φ). Поэтому время, требующееся для распознавания цепочки длиной n , не превосходит Cn^2 .

РЕШЕНИЕ ЗАДАЧИ О ПЕРЕСЕЧЕНИИ НАЧАЛА КООРДИНАТ В РЕАЛЬНОЕ ВРЕМЯ ¹⁾

М. Фишер, А. Розенберг

ВВЕДЕНИЕ

Машины с несколькими лентами или несколькими счетчиками интенсивно изучаются как модели для исследования вычислений с ограничениями во времени. Результаты, связывающие эти две модели, получены Фишером, Мейером и Розенбергом [1]. Известно, например, что любая машина с несколькими счетчиками, работающая в реальное время, может быть промоделирована при любом $\varepsilon > 0$ за время $T(n) = (1 + \varepsilon)n$ на одноленточной машине Тьюринга. Однако еще неизвестно, можно ли положить здесь $\varepsilon = 0$. Мы введем класс языков $\{O_n\}$, каждый из которых распознается в реальное время на n -счетчиковой, но не на $(n - 1)$ -счетчиковой машине. Затем мы докажем довольно удивительный факт, что для всех n язык O_n распознается в реальное время на одноленточной машине Тьюринга. Хотя этот результат не решает по-прежнему открытой проблемы, он показывает трудности, возникающие при любой попытке отрицательно решить эту проблему.

Для краткости мы отсылаем читателя к работе Фишера, Мейера и Розенберга [1] за всеми необходимыми определениями.

1. ЗАДАЧА О ПЕРЕСЕЧЕНИИ НАЧАЛА КООРДИНАТ

Возьмем алфавит $A_n = \{L_1, R_1, \dots, L_n, R_n\}$ из $2n$ букв. Для каждой буквы a из A_n и каждого слова w в A обозначим число вхождений буквы a в w через $(a; w)$.

Язык n -мерного пересечения начала координат ($n = 1, 2, \dots$) — это множество $O_n = \{w \in A_n^* : \text{для всех } 1 \leq i \leq n \ (L_i; w) = (R_i; w)\}$. Геометрический смысл названия станет понятным, если каждую букву $L_i(R_i)$ рассматривать как указание сделать один шаг влево (вправо) параллельно i -й оси в n -мерном пространстве; при этом словам языка O_n соответствуют те и только те пути, которые начинаются и оканчиваются в начале координат.

¹⁾ Fischer M. J., Rosenberg A. L., Real-time solutions of the origin-crossing problem, *Math. Systems Theory*, 2, № 3 (1968), 257—263.

2. РЕШЕНИЕ В КЛАССЕ СЧЕТЧИКОВЫХ МАШИН

Результаты, изложенные в этом разделе, были получены впервые Лэйнгом [3]. Мы приводим их здесь для полноты изложения, а также для того, чтобы дать более простые доказательства.

Теорема 1. При всех $n > 0$ язык O_n (а) распознается в реальное время на n -счетчиковой машине; (б) не распознается в реальное время никакой $(n-1)$ -счетчиковой машиной.

Доказательство. Часть (а) очевидна. Если машина просматривает символ $R_i(L_i)$, в i -й счетчик добавляется $+1$ (-1). Машина воспринимает входное слово в том и только в том случае, когда все n счетчиков содержат нуль.

Чтобы установить (б), заметим, что если язык L распознаваем в реальное время k -счетчиковой машиной, то с тем же числом счетчиков распознаваем язык $L \cap R$, где R — произвольное регулярное множество. (Это очевидно, если вспомнить доказательство теоремы 1 из работы Розенберга [4]). Рассмотрим теперь язык

$$C_n = O_n \cap \{L_1, \dots, L_n\}^* \{R_1, \dots, R_n\}^*.$$

Пусть f — гомоморфизм на A_n , определенный для $i = 1, \dots, n$ так: $f(L_i) = f(R_i) = R_i$. Продолжим f до отображения на A_n^* , положив $f(\lambda) = \lambda$ и $f(a_1, \dots, a_m) = f(a_1) \dots f(a_m)$; λ обозначает пустое слово. Теперь пусть x и y — слова над $\{L_1, \dots, L_n\}$, причем для некоторого i , $1 \leq i \leq n$, $(L_i; x) \neq (L_i; y)$.

Ясно, что, если $(n-1)$ -счетчиковая машина M распознает C_n , она попадает в различные конфигурации после чтения слов x и y , иначе слова $xf(x) \in C_n$ и $yf(x) \notin C_n$ воспринимались бы одинаково. Легко проверить, что число различных конфигураций, которые возникают при работе M со словами длины, не большей k , не менее чем

$$e_k = \sum_{j=0}^k C(j+n-1, n-1),$$

где $C(a, b) = a! / (b!(a-b)!)$. Если M имеет q внутренних состояний, число различных конфигураций, достижимых за k или менее шагов, не превышает $c_k = q(k+1)^{n-1}$. При больших k $e_k > c_k$; полученное противоречие завершает доказательство.

3. РЕШЕНИЕ НА ОДНОЛЕНТОЧНОЙ МАШИНЕ ТЬЮРИНГА

Теорема 2. Для каждого $n > 0$ язык O_n распознается в реальное время на одноленточной машине Тьюринга.

Доказательство. Мы проведем построение требуемой машины T . Разрешим для простоты пишущее-читающей головке

машины просматривать по два квадрата сразу. Переход к требуемой в условии теоремы машине легко произвести, используя хорошо известную теорему об ускорении [2]. T будет моделировать n -счетчиковую машину. При этом T не обязательно перерабатывает входные символы в соответствии с тем, как они поступают на вход, но может на отдельных этапах моделирования откладывать обработку входных символов. Затем небольшие группы сохраненных символов будут всегда перерабатываться как единое целое. Однако если во всех счетчиках стояли малые по модулю числа, таких «сохраненных» символов тоже будет немного, и T сможет распознавать момент восприятия входного слова.

Память T будет состоять из n ограниченных регистров управляющего устройства и рабочей ленты, разделенной на $n + 1$ дорожку. В каждом из регистров будет стоять число r_i , $|r_i| < M = 8n^2 + 2n + 1$ ($1 \leq i \leq n$). Каждая из первых n дорожек ленты разделена на два канала. Первый канал i -й дорожки, $1 \leq i \leq n$, будет содержать двоичное представление целого числа $p_i \geq 0$, другой канал — двоичное представление целого $q_i \geq 0$. Все эти двоичные числа размещены так, что их младшие разряды находятся в одном и том же квадрате ленты; назовем этот квадрат и соседний с ним слева квадрат **домашними квадратами**. Оставшаяся $(n + 1)$ -я дорожка используется для хранения «сохраненных» единиц, пока они не подвергнутся переработке. Каждый квадрат этой дорожки называется **складом** и может содержать любое слово в A_n длины, меньшей или равной $B = 4n + 1$.

Обозначим для любого слова w над A_n $d_i = (R_i; w) - (L_i, w)$, $i = 1, \dots, n$. Прочитав слово w , T будет вычислять приближение c_i к каждому d_i . Это приближение представляется тройкой $\langle p_i, q_i, r_i \rangle$, где

$$c_i = M(p_i - q_i) + r_i.$$

В оставшейся части доказательства мы опишем, как происходит процесс аппроксимации, и покажем, что все $d_i = 0$, $i = 1, \dots, n$, в том и только в том случае, когда для всех i

$$c_i = p_i = q_i = r_i = 0.$$

3.А. Устойчивые конфигурации. Говорят, что конфигурация машины T **устойчива**, если выполняются такие условия:

- (1) Пишущо-читающая головка помещается на домашних квадратах ленты.
- (2) Для каждого $i = 1, \dots, n$ либо (а) $p_i = q_i = 0$, либо (б) $\max(p_i, q_i) = 1$ и $\min(p_i, q_i) = 0$, либо (в) $L(\max(p_i, q_i)) \geq L(\min(p_i, q_i)) + 2$, где $L(x)$ — длина двоичного представления x (без первых нулей).

(3) Ни в одном складе нет слова, длина которого превышала бы удвоенное число нулей и пустых символов, стоящих в каналах квадрата над данным складом.

(4) Начиная с крайнего левого квадрата, у которого хотя бы в одном канале есть единица, во всех квадратах слева от него склады пустые.

(5) Для всех $i = 1, \dots, n$, $|r_i| < M$.

Устойчивая конфигурация называется **чистой**, если для всех $i = 1, \dots, n$ $|r_i| < M/2$.

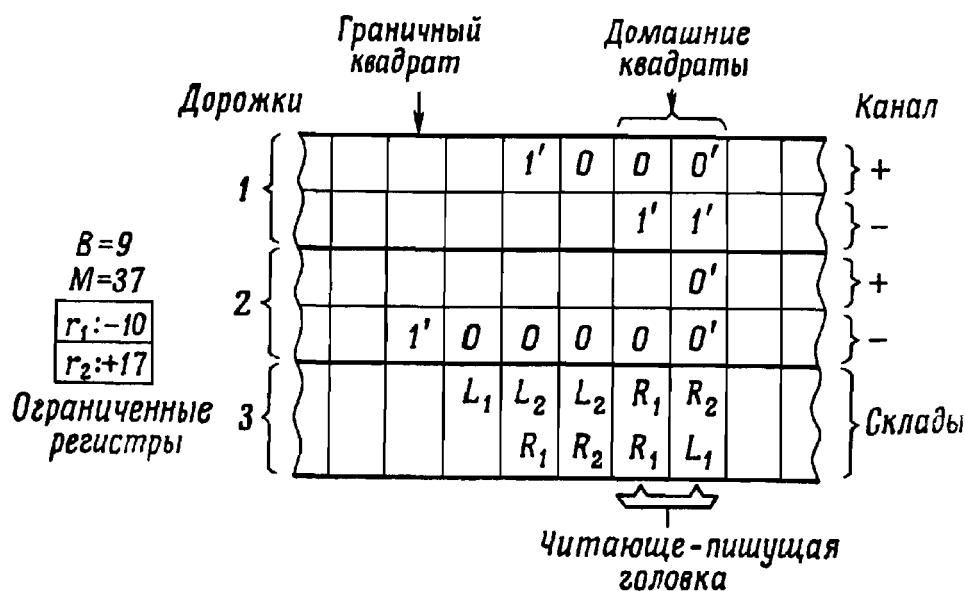


Рис. 1. Чистая устойчивая конфигурация памяти машины T .

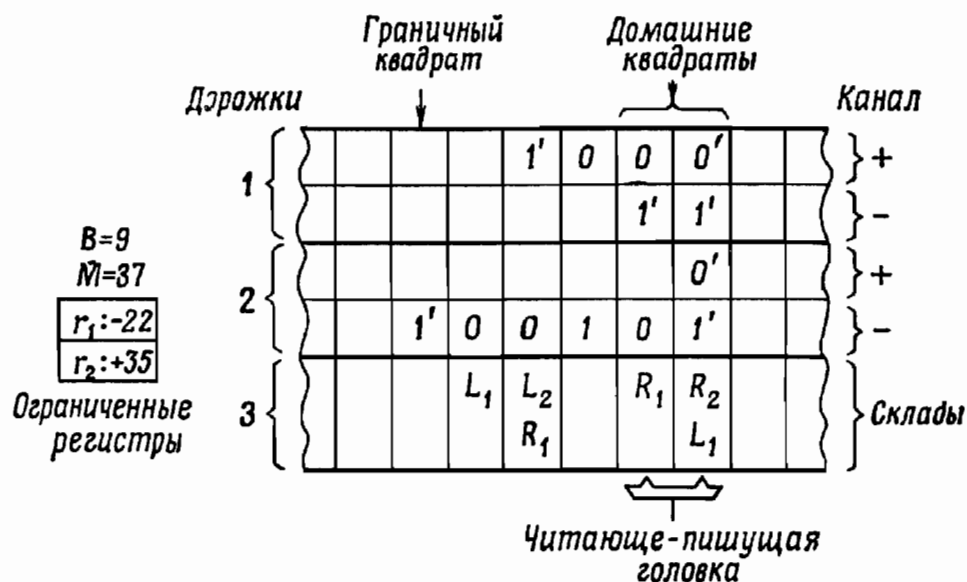
В начале работы T лента пуста, а все $r_i = 0$, поэтому начальная конфигурация является устойчивой и чистой.

Рис. 1 показывает чистую устойчивую конфигурацию в случае $n = 2$. В этом примере $p_1 = 8$, $q_1 = 3$, $p_2 = 0$, $q_2 = 32$, а, значит, $c_1 = 37 \times 5 - 10 = 175$ и $c_2 = 37 \times (-32) + 17 = -1167$, в то время как $d_1 = c_1 + 1 = 176$, а $d_2 = c_2 = -1167$. Цифры старшего и младшего разрядов помечены, чтобы машина T могла распознавать конец каждого канала.

3.В. Фаза исправления. Когда T находится в чистой устойчивой конфигурации, входные символы перерабатываются по мере поступления прибавлением $+1$ или -1 к соответствующему r_i . Как только какое-нибудь из r_i превысит по модулю $M/2$, начинается фаза чистки, которая состоит из исправления регистров и исправления ленты.

Исправление регистров осуществляется прибавлением или вычитанием M из каждого r_i так, чтобы результат заключался в интервале $(-M/2, +M/2)$. Если некоторое r_i было изменено таким образом, то, чтобы оставить величину $c_i = M(p_i - q_i) + r_i$ неизменной, приходится прибавить единицу к p_i или к q_i . Ма-

шина проделывает это, сначала установив, какие именно p_i или q_i следует изменить, а затем производя обычное сложение двоичных чисел одновременно во всех соответствующих каналах. При этом переносы распространяются влево настолько, насколько это необходимо. Каждый входной символ, поступающий на этом этапе вычислений, попадает в склад под правым квадратом пары квадратов, на которой стоит машина в момент его поступления. Если машина заменяет нуль или пустой символ на единицу в каком-либо канале просматриваемого квадрата, то склад под правым квадратом воспринимаемой пары очищается после



Р и с. 2. Устойчивая конфигурация памяти машины T , не являющаяся чистой.

помещения в него текущего входного символа. Если склад, подлежащий очистке, содержит слово u , то для каждого i к r_i прибавляется $(R_i; u) - (L_i; u)$, а затем склад очищается. Поскольку емкости складов ограничены, эту операцию очистки можно провести за один шаг.

Если перенос распространяется до левого конца дорожки (машина узнает это по отмеченному символу), машина пытается вычесть число, записанное в этом канале (которое есть степень 2), из числа в другом канале той же дорожки. Это вычитание выполняется в тех случаях, когда другое число либо имеет ту же длину, что и исправляемое, либо на один разряд больше. Ясно, что для проверки этих условий достаточно просмотреть лишь два квадрата ленты, а затем, возможно, выполнить вычитание.

Закончив описанный проход влево, T возвращается на домашние квадраты и во время возвращения заменяет нули, появившиеся в результате вычисления перед первой слева единицей, пустыми символами. Все входные символы на этом этапе

помещаются в склад квадратов, стоящих справа от просматриваемого.

Рис. 2 иллюстрирует устойчивую конфигурацию машины T , не являющуюся чистой, а рис. 3 — устойчивую конфигурацию, получившуюся после исправления конфигурации на рис. 2. Во время исправления на вход поступило слово $L_1R_2R_1L_2R_1$; q_2 и p_2 были изменены, что вызвало очистку двух складов; на дорожке 1 было произведено вычитание

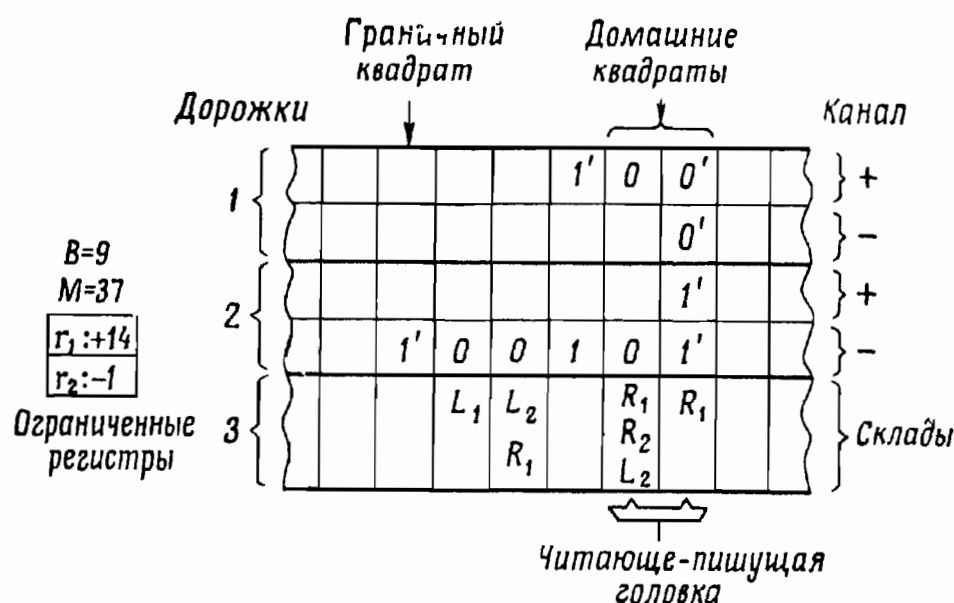


Рис. 3. Конфигурация памяти T после исправления конфигурации на рис. 2.

3.С. Проверка. Сначала проверим, что после очистки T снова находится в устойчивой конфигурации.

Условие (1) выполняется по определению фазы исправления. Предположим, что для каждого i p_i и q_i удовлетворяют условию (2) перед очисткой. Пусть X и Y суть значения соответственно p_i и q_i перед очисткой, а X' и Y' — после очистки. (а) Если ни p_i , ни q_i не исправлялись, условие (2) выполнено. (б) Если p_i (или q_i) исправлялось и, кроме того, производилось вычитание, то X' (или Y') $= 0$ и условие (2) выполнено. (в) Если p_i исправлялось, но вычитания не было, то $X' = X + 1$ и $Y' = Y$. В этом случае либо $L(X') < L(Y') - 1$ и условие (2) выполнено, либо $L(X') > L(Y') = L(Y)$. В последнем случае либо $L(Y') = 0$, либо $L(X') \geq L(X) = L(\max(X, Y)) \geq L(\min(X, Y)) + 2 = L(Y) + 2 = L(Y') + 2$ и снова условие (2) выполнено.

Условие (3) проверяется следующим образом. (а) Если был очищен склад крайнего левого из достигнутых при движении очистки квадратов, то этот склад продолжает оставаться пустым. (б) Если был очищен склад квадрата, не рассмотренного в (а), то в нем стоит единственный символ, поступивший во время движения вправо к домашним квадратам. Но так как склад не на-

ходится на левом конце пути машины, какой-либо перенос проходит над ним и по крайней мере в одном из каналов данного квадрата стоит нуль. (в) Если склад не был очищен, то либо этот квадрат при очистке не посещался машиной, либо машина прошла его дважды и поместила в данный склад два символа. И опять в этом последнем случае, как и в (б), один из каналов данного квадрата содержит нуль, так как какой-либо перенос распространялся над квадратом. Во всех случаях (а) — (в) условие (3) выполняется.

Условие (4), очевидно, выполнено, поскольку вначале все склады пусты, а если в результате вычитания какой-нибудь канал укорачивается, то соответствующий новый конечный склад очищается.

Наконец, предположим, что условие (5) выполнялось перед очисткой. Исправление регистров обеспечивает к началу очистки для всех i условия $|r_i| < M/2$. В процессе любой очистки изменяются не более n каналов, поэтому очищается самое большее n складов. Даже если все очищенные склады затрагивали одно и то же r_i , в конце чистки имеем

$$|r_i| < \frac{1}{2}M + Bn = \frac{1}{2}(2Bn + 1) + Bn = 2Bn + \frac{1}{2} < M,$$

и условие (5) выполняется.

Итак, после фазы очистки машина T находится в устойчивой конфигурации. (Отсюда следует, что ни склады, ни регистры не могут переполняться.)

Наконец, мы покажем, что $p_i = q_i = r_i = 0$, $1 \leq i \leq n$, в том и только в том случае, когда $d_i = 0$, $1 \leq i \leq n$. Сначала положим, что $p_i = q_i = r_i = 0$. Тогда длины всех дорожек равны нулю и по условию (4) необработанных входных символов нет. Следовательно,

$$d_i = c_i = M(p_i - q_i) + r_i = 0, \quad 1 \leq i \leq n.$$

Теперь предположим, что $d_i = 0$, $1 \leq i \leq n$. Пусть b есть длина самого длинного p_i или q_i на ленте.

(а) Если $b = 1$, то $|p_i - q_i| = 1$ и $|c_i| > 0$, поскольку $|r_i| < M$. Однако по условию (4) нет необработанных символов, так что $c_i = d_i = 0$. Это дает вопреки предположению $b \neq 1$.

(б) Если $b \geq 2$, то $|p_i - q_i| \geq 2^{b-2} + 1$ по условию (2). Следовательно, $|c_i| \geq M|p_i - q_i| - |r_i| > M(2^{b-2} + 1) - M = 2^{b-2}M$. В этом случае может быть $(b-1)B$ необработанных символов. Поскольку $n \geq 1$, имеем

$$\begin{aligned} |d_i| &\geq |c_i| - (b-1)B > M \cdot 2^{b-2} - (b-1)B = \\ &= B(2^{b-1}n - b + 1) + 2^{b-2} > 0, \end{aligned}$$

вопреки предположению, что $d_i = 0$. Итак, $b < 2$.

(в) Теперь мы должны заключить, что $b = 0$. В этом случае по условию (4) нет необработанных символов, так что

$$c_i = r_i = d_i = 0, \quad 1 \leq i \leq n.$$

Доказательство окончено.

ЛИТЕРАТУРА

1. Fischer P. C., Meyer A. R., Rosenberg A. L., Counter machines and counter languages, *Math. Syst. Theory*, 2, № 3 (1968), 265—283. (См. стр. 380—400 настоящего сборника.)
2. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Am. Math. Soc.*, 117 (1965), 285—306. (Русский перевод: Хартманис Дж., Стернс Р. Е., О вычислительной сложности алгоритмов, Кибернетический сборник (новая серия), вып. 4, «Мир», (1967).)
3. Laing R., Realization and complexity of commutative events, Univ. Mich Tech. Rep, 03105-48-T, 1967.
4. Rosenberg A. L., Real-time definable languages, *Jour. ACM*, 14 (1967), 645—662. (См. стр. 168—193 настоящего сборника.)

СЧЕТЧИКОВЫЕ МАШИНЫ И СЧЕТЧИКОВЫЕ ЯЗЫКИ ¹⁾

П. Фишер, А. Мейер, А. Розенберг

ВВЕДЕНИЕ

В последние годы возрос интерес к исследованию сложности вычислений на машинах Тьюринга. Параллельно с работой по изучению машин Тьюринга с ограничениями на время и пространство ²⁾ мы изучаем машины с подобными ограничениями, у которых вместо лент используются счетчики. Счетчиковые машины (СМ) вначале кажутся слабее, чем машины Тьюринга (ТМ), но, как показал Минский [8], возможности СМ без ограничений столь же велики, как у ТМ. СМ с ограничениями значительно удобнее для рассмотрения, чем соответствующие ТМ, поэтому многие вопросы, остающиеся открытыми для ТМ с ограничениями, решаются в этой работе для СМ с ограничениями.

Мы будем в основном рассматривать машины — распознаватели языков. Многие свойства ТМ-распознавателей переносятся на СМ, например свойства, связанные с замкнутостью классов языков, распознаваемых в реальное время. С другой стороны, можно продемонстрировать ряд существенных различий между ТМ и СМ; соответствующие свойства ТМ и СМ будут сопоставляться на протяжении всей работы.

Разд. 1 содержит формальное определение СМ-распознавателя. Устанавливается связь между этим определением и его возможными вариантами. В разд. 2 мы исследуем несколько конкретных проблем распознавания, чтобы глубже раскрыть вычислительные возможности СМ. Разд. 3 связан с ограничениями на пространство для СМ. Мы устанавливаем непосредственную связь между временными и пространственными ограничениями на СМ и показываем изоморфизм возникающих в результате пространственных ограничений иерархий у ТМ и СМ. Разд. 4 содержит замечания о времени, которое требуется для моделирования одного вида машины на другом. Наконец, в разд. 5 мы исследуем свойства классов языков, которые определимы СМ

¹⁾ Fischer P. C., Meyer A. R., Rosenberg A. L., Counter machines and counter languages, *Mathematical Systems Theory*, 2, № 3 (1968), 265—283.

²⁾ На длину ленты. — *Прим. ред.*

с различным числом счетчиков. Мы показываем, что эти классы образуют расширяющуюся последовательность, и исследуем некоторые свойства замкнутости.

1. МОДЕЛЬ И ЕЕ ВАРИАНТЫ

1.1. Определения. Алфавит Σ — конечное множество элементов, называемых буквами. Слово над Σ — любая конечная (возможно, пустая) последовательность букв. Σ^* означает множество всех слов в Σ (включая пустое слово λ). Любое подмножество Σ^* называется **языком**. За описанием стандартных операций над словами и языками мы отсылаем читателя к работе Гинзбурга [4].

Односторонняя (on-line) k -счетчиковая машина (k -СМ) состоит из управляющего устройства с конечным числом состояний, k счетчиков, каждый из которых может содержать любое целое число, и входной ленты. Состояния управляющего устройства делятся на **избирательные** и **автономные**. В начале вычислений СМ находится в выделенном начальном состоянии, а во всех счетчиках стоят нули. Шаг СМ однозначно определяется состоянием управляющего устройства, а если оно избирательное, то и считываемым на входной ленте символом, а также множеством счетчиков, которые содержат нуль. Шаг заключается в независимом изменении содержимого каждого из счетчиков путем прибавления 0, +1 или -1 и в изменении состояния управляющего устройства.

Более точно, k -СМ определяется:

(1) конечным непустым множеством Q_p избирательных состояний;

(2) конечным множеством Q_a автономных состояний;

(3) конечным входным алфавитом Σ ;

(4) функцией переходов

$$M: (Q_a \cup (Q_p \times \Sigma)) \times \{0, 1\}^k \rightarrow Q_a \cup Q_p;$$

(5) функцией изменения счетчиков

$$K: (Q_a \cup (Q_p \times \Sigma)) \times \{0, 1\}^k \rightarrow \{-1, 0, 1\}^k;$$

(6) выделенным начальным состоянием s_0 в Q_p ;

(7) выделенным подмножеством $F \subseteq Q_p$ конечных состояний.

Пусть Z означает множество целых чисел и Z^k — множество всех наборов длины k из этих чисел. Для любого целого x определим функцию

$$\text{sg}(x) = \begin{cases} 0, & \text{если } x = 0, \\ 1, & \text{если } x \neq 0, \end{cases}$$

и продолжим sg на Z^k , положив

$$sg(\langle x_1, \dots, x_k \rangle) = \langle sg(x_1), \dots, sg(x_k) \rangle.$$

Конфигурацией k -СМ назовем элемент множества $(Q_a \cup Q_p) \times \Sigma^* \times Z^k$. Мы пишем

$$(q, a\omega, \langle x_1, \dots, x_k \rangle) \rightarrow (q'', \omega, \langle y_1, \dots, y_k \rangle)$$

или

$$(q', \omega, \langle x_1, \dots, x_k \rangle) \rightarrow (q'', \omega, \langle z_1, \dots, z_k \rangle),$$

где $q \in Q_p$, $q' \in Q_a$, $a \in \Sigma$, $\omega \in \Sigma^*$, $x_i, y_i, z_i \in Z$, если

$$(1) \quad M(q, a, sg(\langle x_1, \dots, x_k \rangle)) = q''$$

и

$$\langle x_1, \dots, x_k \rangle + K(q, a, sg(\langle x_1, \dots, x_k \rangle)) = \langle y_1, \dots, y_k \rangle$$

или

$$(2) \quad M(q', sg(\langle x_1, \dots, x_k \rangle)) = q''$$

и

$$\langle x_1, \dots, x_k \rangle + K(q', sg(\langle x_1, \dots, x_k \rangle)) = \langle z_1, \dots, z_k \rangle.$$

Для конфигураций C и C' мы напишем $C \Rightarrow C'$, если либо $C = C'$, либо существуют конфигурации $C = C_0$, $C_1, \dots, C_t = C'$, такие, что $C_i \rightarrow C_{i+1}$ для каждого i , $0 \leq i < t$.

k -СМ останавливается через t шагов при вычислении на слове ω , если существуют конфигурации $C_0, C_1, \dots, C_t$, такие, что

$$(s_0, \omega, \langle 0, \dots, 0 \rangle) = C_0 \rightarrow C_1 \rightarrow \dots \rightarrow C_t = (q, \lambda, \langle x_1, \dots, x_k \rangle)$$

(здесь $q \in Q_p$). k -СМ **допускает (принимает)** или **отвергает** ω , смотря по тому, где содержится q : в F или в $Q_p \setminus F$ соответственно.

Пространство, необходимое СМ при работе с ω , — это сумма максимумов абсолютных значений содержимых каждого из счетчиков во время t шагов вычислений.

Пусть T и S — монотонно возрастающие функции. СМ работает **время T (на пространстве S)**, если для всех $n \geq 0$ СМ, начиная работу с входным словом ω длины n , останавливается через $T(n)$ или менее шагов (требует пространство, не превышающее $S(n)$).

Язык L в Σ **распознаваем** СМ, если СМ принимает каждое слово из L и отвергает каждое слово из $\Sigma^* \setminus L$. Язык L **распознается за время $T(n)$ (на пространстве $S(n)$)**, сокращенно, $T(n)$ -распознаваем ($S(n)$ -распознаваем), если существует СМ, распознающая L , которая работает время T (на пространстве S).

Мы выделяем случай $T(n) = n$. Язык, который распознаваем за время $T(n) = n$, называется **распознаваемым в реальное время**.

Значение этого случая определяется важностью поиска эффективных алгоритмов синтаксического анализа.

По аналогии с нашими определениями можно определить соответствующие понятия для многоленточных ТМ. Для краткости мы отсылаем читателя за подходящими определениями и связанными с ними результатами к следующим статьям: Рабин [11], Хартманис и Стирнз [6], Стирнз, Хартманис, Льюис [14] и Розенберг [13].

1.2. Варианты модели. СМ определяются в литературе разными способами. Стоит проверить для случая временных ограничений, что наша модель не чувствительна к слабым изменениям условий. Говорят, что две машины *эквивалентны по времени*, если обе распознают один и тот же язык и если одна работает в течение времени $T(n)$, то и другая работает в течение такого же времени.

Теорема 1.1. *Для любой k -СМ, которая может за один шаг независимо изменять содержимое каждого из счетчиков на любое целое число, заключенное между $+c$ и $-c$ (c фиксировано), можно эффективно построить эквивалентную по времени (обычную) k -СМ.*

Доказательство. Мы дадим набросок конструкции, оставив детали читателю. Когда счетчики исходной k -СМ содержат числа $m_1, \dots, m_k$, счетчики «сжатой» k -СМ содержат $[m_1/c], \dots, [m_k/c]$, сохраняя (ограниченные) остатки $m_1 - [m_1/c]c, \dots, m_k - [m_k/c]c$ в памяти с конечным числом состояний¹⁾. Ясно, что счетчики «сжатой» машины изменяются за шаг не более чем на единицу.

Теорема 1.2. *Для любой k -СМ можно эффективно построить эквивалентную по времени k -СМ, которая (а) хранит в счетчиках только неотрицательные числа; (б) изменяет содержимое счетчиков не чаще, чем через каждые c шагов ($c > 0$); (в) изменяет за один шаг содержимое не более одного счетчика.*

Доказательство. Сохраняя абсолютные значения чисел в счетчиках, а их знаки — в памяти управляющего устройства (число состояний которого придется увеличить в 2^k раз), мы докажем (а). Для доказательства (б) и (в) используется техника «сжатия» из теоремы 1.1.

Начиная с этого места, мы молчаливо предполагаем, что счетчики содержат только неотрицательные числа, т. е. выполняется условие (а).

¹⁾ $[x]$ означает целую часть x .

В отличие от этих результатов, устанавливающих инвариантность, мы покажем, что наличие входной ленты с двусторонним движением сильно изменяет минимум времени вычислений. **Двусторонняя** (off-line) модель имеет входную ленту, на которой помещается только читающая головка. Входное слово w записывается на входной ленте в виде $\$w\$$, и читающая головка помещается вначале на крайний левый символ слова w (или на правое $\$$, если $w=\lambda$). На каждом шаге двусторонняя СМ может передвигать читающую головку на один квадрат влево или вправо по входной ленте, не выходя за границы области, в которой записано входное слово. Под принимающими и отвергающими состояниями понимают состояния **остановки**, и слово принимается или отвергается согласно состоянию, в котором останавливается СМ. За более подробными сведениями о двусторонних машинах отсылаем читателя к статье Стирнза, Хартманиса и Льюиса [14].

Легко проверить, что (1) различие между одно- и двусторонними СМ несущественно в случае реального времени и (2) для данной односторонней СМ можно эффективно указать эквивалентную по времени двустороннюю СМ.

Чтобы сопоставить одно- и двусторонние СМ, рассмотрим язык $L = \{xax^T \mid x \in \{0, 1\}^*, a \notin \{0, 1\}\}$, где x^T — обращение слова x .

Теорема 1.3. (а) Язык L не распознается никакой односторонней k -СМ, которая работает время, меньшее чем $T(n) = 2^{n/2k}$; (б) распознается двусторонней СМ, которая работает время $T(n) = cn^2/\log n$ для некоторой постоянной c ; (в) каждая СМ, распознающая L , работает в течение времени $T(n) \geqslant \geqslant cn^2/\log n$ для некоторой константы c и для всех n .

Доказательство. (а) **Конфигурация памяти** односторонней k -СМ — это $(k+1)$ -набор, состоящий из текущего внутреннего состояния и содержимого всех k -счетчиков. Любая односторонняя k -СМ, распознающая L , должна иметь различные конфигурации памяти после чтения различных двоичных слов x и y , в противном случае xax^T и yax^T были бы восприняты одинаково. Пусть t есть максимум содержимого всех счетчиков после чтения слова длины m . Если k -СМ имеет q состояний, то число различных конфигураций памяти k -СМ после чтения всех слов длины m не превышает $q \cdot (t+1)^k$. В силу предыдущих замечаний $q \cdot (t+1)^k \geqslant 2^m$ и, значит, $t \geqslant c \cdot 2^{m/k}$ для некоторой постоянной c . Поскольку k -СМ может изменять за один шаг содержимое не более чем одного счетчика, она должна сделать не менее $c \cdot 2^{(n-1)/2k} \geqslant d \cdot 2^{n/2k}$ шагов (при некотором d), чтобы работать с входными словами xax^T длины n . Утверждение (а) доказано.

(б) Мы кратко опишем работу двусторонней СМ M , которая распознает L .

(1) M проходит по слову x , запоминая в счетчике $\lceil \log m \rceil$, где m — длина x . (В этой работе все логарифмы берутся по основанию 2.)

(2) M последовательно выбирает $\lceil \log m \rceil$ двоичных символов из x , рассматривая последовательность $b_n, b_{n-1}, \dots, b_0$ как двоичную запись целого числа $\sum_{i=0}^n (b_i + 1) 2^i$, затем M выбирает соответствующие $\lceil \log m \rceil$ символов y и проверяет, представляют ли они то же число. Этот шаг очевидным образом изменяется, когда машина встречает «а».

Ясно, что (1) требует m шагов. На шаге (2) для обращения записи числа требует $d \cdot 2^{\lceil \log m \rceil} \leq d \cdot m$ операций, где d — некоторая константа. Аналогично нужно время, пропорциональное m , чтобы читающая головка заняла новое положение (перешла со слова x на слово y или наоборот). Таким образом, выполнение шага (2) требует $e \cdot m$ шагов с некоторой константой e . Шаг (1) выполняется один раз, а шаг (2) не более $m/\lceil \log m \rceil + 1$ раз. Следовательно, M распознает L за время

$$T(n) \leq m + e \cdot m (m/\lceil \log m \rceil + 1) \leq c \cdot n^2 \lceil \log n \rceil.$$

(в) Чтобы показать, что время работы M быть может лишь в константу раз превосходит наилучшее возможное время, мы переформулируем результат Кобхема [1].

Лемма. Для любой СМ, распознающей язык L , имеем

$$(1) \quad \inf_{n \rightarrow \infty} \frac{\log S(n)}{\log n} > 0,$$

$$(2) \quad \inf_{n \rightarrow \infty} \frac{T(n) \cdot \log S(n)}{n^2} > 0.$$

В разд. 3 мы покажем, что для некоторого положительного c условие (1) влечет $[S(n)]^c > T(n)$ для некоторой константы $c > 0$, откуда для всех n и некоторого постоянного d получаем из (2)

$$(*) \quad T(n) \cdot (\log T(n)) \geq d \cdot n^2.$$

Из этого следует, что

$$T(n) \geq \frac{d}{2} \frac{n^2}{\log n},$$

что справедливо при всех n , как легко убедиться. Доказательство теоремы 1.3 окончено.

2. НЕКОТОРЫЕ ЧАСТНЫЕ ПРОБЛЕМЫ РАСПОЗНАВАНИЯ

Цель этого раздела — глубже понять возможности СМ. Замечания о том, что СМ могут решать те задачи, которые формулируются в терминах пересчета, является тавтологией. Однако вызывает удивление широта класса таких задач: как показал Минский [8], при отсутствии временных ограничений любая (частично) вычислимая проблема может быть сформулирована как проблема пересчета¹⁾.

Мы приведем сейчас три проблемы, которые могут быть решены на СМ за реальное время. В этом разделе вместо доказательств мы ссылаемся на соответствующие источники.

2.1. Арифметические выражения. Первый из приводимых нами языков состоит из правильно построенных арифметических выражений без скобок.

Рассматривая лишь одно- и двуместные операции, можно определить язык L_2 (нижний индекс означает максимальную «местность» операций) такой грамматикой:

$$\begin{aligned} \langle \text{выражение} \rangle &\rightarrow \langle \text{одноместный оператор} \rangle \langle \text{выражение} \rangle, \\ \langle \text{выражение} \rangle &\rightarrow \langle \text{двуместный оператор} \rangle \langle \text{выражение} \rangle \langle \text{выражение} \rangle, \\ \langle \text{выражение} \rangle &\rightarrow \langle \text{операнд} \rangle. \end{aligned}$$

Читатель легко сам определит язык L_n в общем случае.

Теорема 2.1. Для всех n язык L_n распознаваем за реальное время с помощью 1-СМ.

Доказательство следует из алгоритма, предложенного Эттингером [9]. Каждому символу алфавита сопоставляется числовое значение: -1 каждому операнду и $+(n-1)$ каждому n -арному оператору ($n \geq 1$). 1-СМ читает входное слово, складывая в счетчике числовые значения читаемых символов. Слово допускается в том и только в том случае, когда (1) после его прочтения счетчик содержит -1 и (2) ни в один из предыдущих моментов вычислений в счетчике не было отрицательного числа. Легко проверить, что СМ распознает L_n . Теперь теорема следует из теоремы 1.1.

Итак, даже 1-СМ может выполнять в реальное время довольно сложные вычисления.

2.2. Коммутативные языки. Язык L называется *коммутативным*, если все перестановки символов любого слова из L также принадлежат L . Ясно, что при распознавании коммутативного

¹⁾ Это значит, что любая частично рекурсивная функция может быть вычислена на СМ. — Прим. ред.

языка достаточно рассматривать лишь число вхождений различных символов во входное слово.

Для данного алфавита $\Sigma = \{a_1, \dots, a_n\}$ определим для любого слова w в Σ $\otimes(w)$ как n -ку отрицательных чисел $\langle \otimes_1(w), \dots, \otimes_n(w) \rangle$, где $\otimes_i(w)$, $1 \leq i \leq n$, — число вхождений a_i в w .

Множество S n -ок целых чисел называется **линейным**, если существуют такие n -ки неотрицательных целых чисел, $c, p_1, \dots, p_r$, что

$$S = \left\{ c + \sum_{i=1}^r k_i p_i \mid \text{все } k_i \geq 0 \right\}.$$

S называется **полулинейным**, если оно является конечным объединением линейных множеств.

Используя результаты Гинзбурга и Спеньера [5] и применяя правило Крамера для решения систем линейных уравнений, можно показать, что для любого полулинейного множества n -ок целых чисел S и для любого n -буквенного алфавита $\Sigma = \{a_1, \dots, a_n\}$ язык $\{w \in \Sigma^* \mid \otimes(w) \in S\}$ есть конечная булевская комбинация множеств, которые распознаются 1-СМ в реальное время. Обратно, поскольку 1-СМ в реальное время распознает лишь контекстно-свободные языки (см. Фишер [2]), из теоремы Парика [10] следует, что для любого языка L , распознаваемого в реальное время с помощью 1-СМ, множество $\{\otimes(w) \mid w \in L\}$ полулинейное. Мы установили таким образом результат, полученный в слегка измененном виде Лэйнгом [7].

Теорема 2.2 (Лэйнг). Пусть S — множество n -ок целых чисел, а $\Sigma = \{a_1, \dots, a_n\}$. Язык L в Σ , равный $\{w \in \Sigma^* \mid \otimes w \in S\}$, есть конечная булевская комбинация языков, распознаваемых 1-СМ в реальное время, в том и только в том случае, когда S — полулинейное множество.

Два примера коммутативных языков представляют особый интерес и заслуживают отдельного рассмотрения.

2.3. Пути в n -мерном пространстве. Легко показать, что для данной n -СМ, работающей в реальное время, можно найти эквивалентную по времени n -ленточную ТМ. Имея в виду, что ленты ТМ обладают большей гибкостью при вычислениях, естественно спросить может ли ТМ с менее чем n лентами в реальное время моделировать n -СМ, также работающую реальное время. В одном из рассматриваемых нами случаев этот вопрос открыт, в другом недавно решен в положительном смысле.

Пусть $\Sigma_n = \{a_1, \dots, a_{2n-1}, a_{2n}\}$ — алфавит из $2n$ букв.

n -мерная проблема пересечения осей — это язык

$$A_n = \{w \in \Sigma^* \mid \text{для некоторого } i, 1 \leq i \leq n, \otimes_{2i-1}(w) = \otimes_{2i}(w)\}.$$

Открытая проблема. Очевидно, что для всех n A_n распознается в реальное время на n -СМ. Открытым является вопрос, распознается ли в реальное время язык A_2 с помощью 1-ТМ.

n -мерная проблема пересечения начала (координат) связана с языком

$$O_n = \{\omega \in \Sigma_n^* \mid \text{для всех } i, 1 \leq i \leq n, \otimes_{2i-1}(\omega) = \otimes_{2i}(\omega)\}.$$

Геометрический смысл названий становится понятным при интерпретации каждой буквы a_{2i-1} (a_{2i}) как указания сделать один шаг влево (вправо) вдоль i -й оси n -мерного пространства.

В отличие от открытой проблемы для языка A_n в случае O_n имеется следующий результат М. Фишера и Розенберга [3].

Теорема 2.3. Для всех $n \geq 1$ язык O_n распознаваем в реальное время 1-ленточной ТМ.

Следствие. Пусть $\Sigma = \{a_1, \dots, a_n\}$. Язык $\{x\beta y \mid x, y \in \Sigma^*, \beta \notin \Sigma, \text{ а } x \text{ — перестановка } y\}$ и язык $\{\omega \in \Sigma^* \mid \otimes_1(\omega) = \dots = \otimes_n(\omega)\}$ распознаваемы в реальное время с помощью 1-ТМ

3. ПРОСТРАНСТВЕННЫЕ ОГРАНИЧЕНИЯ

3.1. Классы пространственной сложности. Стернз, Хартманис и Льюис [14] определили для данной функции S **класс ленточной сложности** C_S как класс языков, распознаваемых ТМ, работающими на ленте длины $\leq S$. Классы ленточной сложности частично упорядочены теоретико-множественным включением. Они имеют богатую структуру, в частности существует бесконечно много несравнимых бесконечных цепей. Если определить по аналогии класс C'_S языков, распознаваемых с помощью СМ на пространстве S , можно получить такую же структуру. В этом разделе мы исследуем классы C'_S и устанавливаем основной результат: $C'_S = C_{\log S}$ для любой функции S .

Сначала мы продемонстрируем два результата, относящихся к «сжатию» пространства; тот факт, что это сжатие можно произвести *без потери времени*, будет полезен в следующих разделах.

Лемма 3.1. Для любой k -СМ, работающей на пространстве $S(n)$, и любого $c > 0$ можно найти эквивалентную по времени k -СМ, работающую на пространстве $[c \cdot S(n)]$.

Лемма следует из теоремы 1.2(б); она может быть усилена с помощью добавления к СМ счетчиков.

Лемма 3.2. Пусть S — возрастающая функция. Для любой СМ, работающей на пространстве, ограниченном полиномом от

S, можно найти эквивалентную по времени СМ, которая работает на пространстве *S*.

Доказательство. Мы опишем, как любой счетчик СМ можно заменить двумя счетчиками, содержимое которых ограничено квадратным корнем содержимого исходного счетчика. Повторив эту конструкцию несколько раз, мы получим требуемый результат.

Процесс моделирования протекает двумя различными способами, причем текущий способ запоминается в конечной памяти. Способ 1 соответствует случаю, когда содержимое моделируемого счетчика есть число $m = r^2 + 2i$ ($0 \leq i \leq r$), способ 2 — случаю $m = r^2 + 2i + 1$ ($0 \leq i < r$).

Способ 1. Заменяем счетчик двумя счетчиками *A* и *B*, содержащими $r - i$ и i соответственно. Если к содержимому исходного счетчика прибавляется $+1$, мы различаем два случая. Если $i = r$, т. е. *A* содержит 0, мы прибавляем $+1$ в счетчик *B*, меняем ролями счетчики *A* и *B* и продолжаем работать способом 1. Если же $i \neq r$, содержимое счетчиков не меняем, но переходим к способу 2.

Способ 2. Снова счетчик *A* содержит $r - i$, а счетчик *B* число i . В случае прибавления $+1$ в исходный счетчик мы прибавляем $+1$ в *B*, -1 в *A* и переходим к способу 1.

Чтобы промоделировать уменьшение содержимого исходного счетчика, обращаем процесс с очевидными изменениями. Ясно, что в исходном счетчике будет нуль тогда и только тогда, когда и в *A* и в *B* нули. Аналогично, когда исходный счетчик содержит $m \geq 0$, один из счетчиков, *A* или *B*, содержит $i = [(m - [\sqrt{m}]^2)/2] \leq [\sqrt{m}]$, а другой $[\sqrt{m}] - i \leq [\sqrt{m}]$. Наконец, наша новая СМ для моделирования одного шага требует также один шаг.

Используя леммы 3.1 и 3.2, мы можем доказать основной результат этого раздела, а именно что ТМ и СМ имеют изоморфные иерархии пространственной сложности.

Теорема 3.1. *Язык распознаваем СМ, работающей на пространстве S , в том и только в том случае, когда он распознаваем ТМ, работающей на пространстве $\log S$.*

Доказательство. Кодирова содержимое счетчиков СМ двоичными записями и заноса эти записи (с возможными сокращениями) на отдельные дорожки ленты ТМ, можно получить ТМ, работающую на пространстве $\log S$ и моделирующую данную СМ.

Чтобы показать обратное, используем алгоритм, предложенный Фишером [2] для моделирования ленты машины Тьюринга с помощью трех счетчиков *A*, *B* и *C*. Предположим, что ТМ

имеет в рабочем алфавите m различных символов: 0 (пустой), 1, ..., $m-1$. Если в некоторый момент непустая часть ленты ТМ имеет вид $a_r a_{r-1} \dots a_0 b_0 b_1 \dots b_s$, а пишущее-читающая головка стоит в s , тогда счетчик A содержит число

$$|a_r| \cdot m^r + |a_{r-1}| \cdot m^{r-1} + \dots + |a_1| \cdot m + |a_0|,$$

счетчик B — число

$$|b_s| \cdot m^s + |b_{s-1}| \cdot m^{s-1} + \dots + |b_1| \cdot m + |b_0|,$$

а счетчик C — нуль ($|a_k|$ означает номер символа a_k в вышеприведенном списке). Число $|c|$ хранится в конечной памяти.

Если ТМ заменяет s на d и передвигает головку вправо, СМ умножает содержимое счетчика A на m и прибавляет $|d|$, затем делит содержимое B на m , запоминая остаток $|b_0|$ в конечной памяти. Передвижение влево моделируется аналогично. В процессе работы счетчик C используется как вспомогательный регистр.

Теперь ясно, что если ТМ просматривает не более $\lceil \log S \rceil$ квадратов ленты, то содержимое счетчиков A , B , C никогда не превосходит

$$m^{\lceil \log S \rceil + 1} \leq m \cdot S^{\lceil \log m \rceil + 1}.$$

Результат следует теперь из леммы 3.2.

3.2. Связь пространственных и временных ограничений. Существует сильная зависимость между временем и пространством при вычислениях на СМ, точно выражаемая следующей теоремой.

Теорема 3.2. Пусть S — функция, $S(n) \geq n$. Язык L распознаваем СМ, которая работает на пространстве S , в том и только в том случае, когда он распознаваем СМ, время работы которой ограничено полиномом от S .

Доказательство. Пусть L распознается СМ, которая имеет q состояний, k счетчиков и работает на пространстве S . При работе со входом длины n может возникнуть не более $q \cdot n (S(n) + 1)^k$ различных конфигураций (определение конфигурации см. в разд. 1.1). Ясно, что машина никогда не остановится, если в процессе вычислений какая-то конфигурация повторяется. Но по определению при распознавании СМ всегда останавливается, поэтому время работы СМ оценивается как

$$T(n) = q \cdot n \cdot (S(n) + 1)^k \leq c \cdot S(n)^{k+1}$$

для некоторой не зависящей от n константы c . Обратно, если время работы ограничено полиномом от S , то и пространство

ограничено тем же полиномом, поскольку на каждом шаге не более чем один счетчик может увеличить свое содержимое. Для окончания доказательства остается прибегнуть к лемме 3.2.

Открытая проблема. Справедливо ли аналогичное теореме 3.2 утверждение для ТМ?

3.3. Классы сложности. Стирнз, Хартманис и Льюис [14] заметили, что при изучении произвольного класса пространственной сложности C_S в случае ТМ достаточно рассматривать одноленточные ТМ. Мы покажем, что для СМ это неверно, в частности класс языков, распознаваемых $(k+1)$ -СМ, работающими на пространстве $S(n) = n$, включает в собственном смысле класс языков, распознаваемых k -СМ, работающими на пространстве $S(n) = n$, и это утверждение справедливо при всех k .

Пусть для $k = 1, 2, \dots$ $L_k = \{0^{m_1}10^{m_2}1 \dots 0^{m_k}\beta_i 0^{m_i} \mid 1 \leq i \leq k, \text{ при всех } i, j \beta_i \notin \{0, 1\}, m_j \geq 1\}$.

Теорема 3.3. (а) При каждом k язык L_k распознаваем в реальное время при помощи k -СМ на пространстве $S(n) = n$. (б) Если r -СМ M распознает L_k для некоторого $k \geq r$, то M работает на пространстве $S(n) \geq cn^{k/r}$ (c — некоторая константа).

Доказательство. Непосредственная проверка (а) представляется читателю.

Чтобы установить (б), заметим, что различные двоичные слова вида $0^{m_1}1 \dots 0^{m_k}1$ должны переводить M в различные конфигурации памяти. Если все $m_j \leq h$, число таких различных слов не превышает h^k . Если максимум содержимого счетчиков при работе с этими словами не превышает t , то число различных конфигураций памяти M не более чем $q(t+1)^r$, где q — число состояний M . Должно быть $q(t+1)^r \geq h^k$, откуда $t \geq d \cdot h^{k/r}$ для некоторой константы d . Но $h+1 \geq n/(k+1)$, где n — длина входного слова. Следовательно, M работает на пространстве $S(n) \geq c \cdot n^{k/r}$ (c — некоторая константа).

Следствие. Для всех k класс языков, распознаваемых $(k+1)$ -СМ на пространстве $S(n) = n$, собственно включен в соответствующий класс для k -СМ.

4. МОДЕЛИРОВАНИЕ ОДНОЙ МАШИНЫ НА ДРУГОЙ

В этом разделе мы исследуем точные границы времени, необходимого для моделирования одной машины с помощью другой. Для простоты мы ограничимся рассмотрением моделирования машин, которые работают линейное время, т. е. $T(n) = cn$ для некоторой константы c . Несколько результатов немедленно обобщаются на случай произвольного $T(n)$.

4.1. Машины Тьюринга и счетчиковые машины. Очевидно, что для любой временной функции $T(n)$ и данной k -СМ, работающей время $T(n)$, можно найти эквивалентную по времени k -ТМ. Довольно удивительно, однако, что 1-ТМ может моделировать работающую реальное время k -СМ с небольшой потерей времени, а k -СМ, работающую время $T(n) > (1 + \varepsilon)n$, — вообще без потери времени.

Теорема 4.1. Для любой k -СМ, работающий линейное время $T(n) \leq cn$, и любого $\varepsilon > 0$ можно эффективно построить эквивалентную одноленточную ТМ, которая работает время $T(n) = (1 + \varepsilon)n$.

Доказательство протекает в три этапа. (а) Для данной k -СМ M находим эквивалентную по времени k -СМ M' , которая изменяет за шаг содержимое не более чем одного счетчика. (б) Затем конструируем одноленточную ТМ T , эквивалентную M' , которая работает время $T(n) = 12cn$. (в) Наконец, для любого числа $d > 0$ строим одноленточную ТМ T' , эквивалентную T , которая работает время $T(n) = (1 + (12c - 1)d)n$. Поскольку возможность шагов (а) и (в) показана в теореме 1.2 и в теореме ускорения Хартманиса и Стирнза [6], мы рассмотрим только шаг (б).

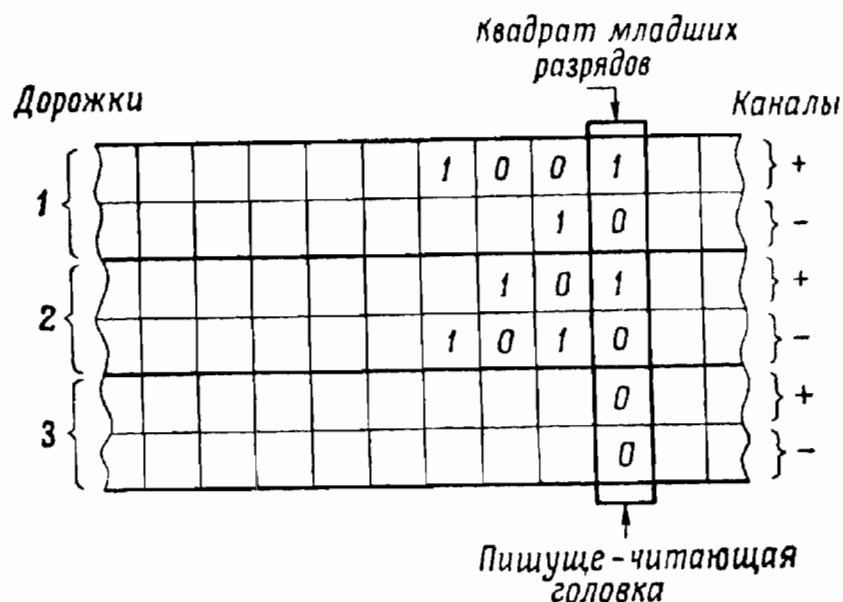


Рис. 1. Возможная конфигурация ленты машины T после 15 шагов машины M' . Счетчик 1 содержит 7, счетчик 2 содержит -5 и счетчик 3 содержит 0.

Для данной машины M' построим T следующим образом. Лента T разделена на k дорожек, по одной для каждого счетчика M , а каждая дорожка разделена на два канала. На каждой дорожке появляется неотрицательное целое число в двоичном представлении; эти числа размещены на дорожках так, что их младшие разряды всегда находятся в одном и том же квад-

рате ленты (рис. 1). Если i -й счетчик M' на n -м шаге вычислений содержит число x_i ($i=1, \dots, k$), то на n -м шаге моделирования i -я дорожка ленты T будет содержать числа y_i и z_i со свойствами: (1) $y_i - z_i = x_i$, (2) $y_i + z_i \leq n$ и (3) нет разряда, в котором представления обоих чисел содержали бы единицу. Последнее условие означает, что $x_i = 0$ тогда и только тогда, когда $y_i = z_i = 0$.

В случае $k=3$ рис. 1 иллюстрирует возможную конфигурацию ленты T после 15 шагов M' . Счетчики M' содержат соответственно 7, -5 и 0.

Процесс моделирования протекает так. Если в i -й счетчик прибавляется (вычитается) единица, то она же прибавляется к y_i (z_i). После выполнения необходимых переносов T передвигается еще на один квадрат влево, чтобы определить, достигнута ли самая левая для каналов дорожки i цифра. Затем T выполняет «чистку», именно движется вправо вплоть до пустого квадрата, следующего за квадратом младшего разряда, и во время этого движения проверяет условие (3), производя необходимые исправления, и, кроме того, заменяет нули, появляющиеся в данном канале слева от крайней единицы, пустыми символами. Ясно, что во время «чистки» T проверяет также значения счетчиков на равенство нулю. И наконец, после «чистки» T передвигается на один квадрат влево, занимая снова квадрат младшего разряда.

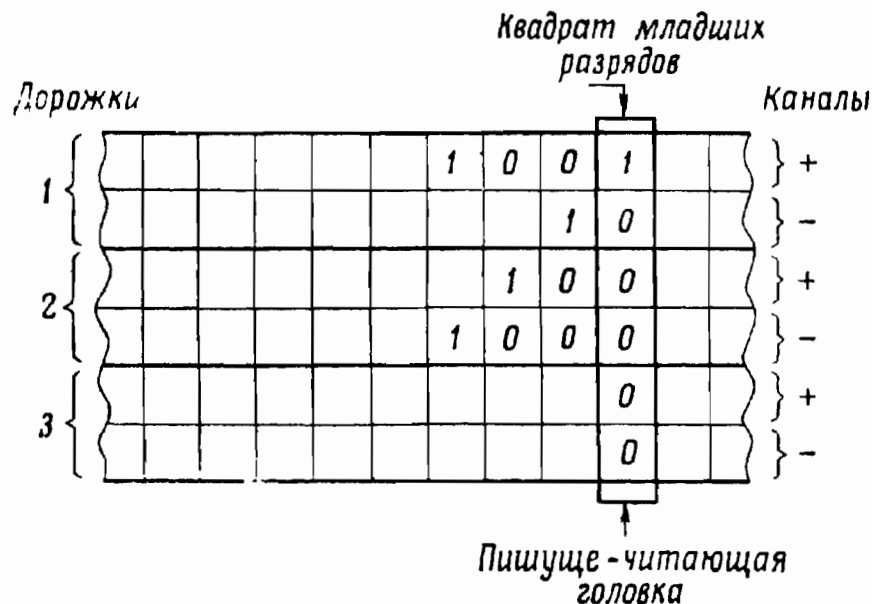


Рис. 2. Конфигурация ленты машины T после моделирования прибавления единицы к содержимому счетчика 2 машины M' .

Рис. 2 показывает изменение конфигурации ленты T , происходящее в результате прибавления единицы во второй счетчик M' , причем M' перед этим находилась в конфигурации рис. 1.

Чтобы оценить полное время моделирования M' с помощью T , заметим, что при моделировании первых 2^{h+1} шагов вычислений

M' самым худшим для T является случай, когда возрастает содержимое одного и того же счетчика M' . В этом случае будет 2^h шагов, когда переноса не требуется, 2^{h-1} шагов, включающих перенос на 1, 2^{h-2} шагов, включающих перенос на 2, ..., 1 шаг с переносом на h и 1 с переносом на $h+1$. Поскольку моделирование шага M' , включающего перенос длины i , требует $2i+4$ шагов T , мы видим, что T затрачивает время

$$\begin{aligned} T(2^{h+1}) &= \sum_{i=0}^h 2^{h-i} (2i+4) + 2(h+1) + 4 = \\ &= \sum_{i=0}^h 2^{h-i} (2i+4) + \sum_{i=h+1}^{\infty} 2^{h-i} (2(h+1)+4) \leq \\ &\leq \sum_{i=0}^{\infty} 2^{h-i} (2i+4) = \\ &= 2^h \left(2 \sum_{i=0}^{\infty} i 2^{-i} + 4 \sum_{i=0}^{\infty} 2^{-i} \right) = \\ &= 2^h \cdot 12. \end{aligned}$$

Теперь для данного n выберем $h = \lceil \log n \rceil$ так, что $2^h \leq n < 2^{h+1}$. Тогда $T(n) \leq T(2^{h+1}) \leq 12 \cdot 2^h \leq 12n$. (Подробный анализ показал бы, что в действительности верхняя оценка $6n$.) Итак, теорема 4.1 доказана.

Открытая проблема. Можно ли положить ε в теореме 4.1 равным 0? Эту задачу не удастся решить даже для 2-СМ в реальное время (см. проблему пересечений осей в разд. 2.3).

В отличие от ситуации в теореме 4.1 односторонней СМ может потребоваться экспоненциальное время для моделирования одноленточной ТМ, работающей реальное время.

Теорема 4.2. (а) Для данной k -ленточной ТМ, работающей линейное время, можно найти эквивалентную одностороннюю СМ, которая работает время $T(n) = a^n$ для некоторого постоянного a . (б) Существует язык L , который распознаваем в реальное время на одноленточной ТМ, но который не распознаваем никакой k -СМ, работающей время меньшее, чем $T(n) = 2^{n/2k}$.

Доказательство. ТМ работает линейное время на пространстве $S(n) = cn$, утверждение (а) теперь следует из теорем 3.1 и 3.2.

Язык $L = \{xax^T \mid x \in \{0, 1\}^*, a \notin \{0, 1\}\}$, как легко показать, распознается в реальное время одноленточной ТМ (в действительности даже pushdown-автоматом в реальное время). Поэтому (б) выводится из доказательства теоремы 1.3(а).

Время, которое требуется СМ для моделирования ТМ, работающей линейное время, оценивается как

$$c_1^n \leq T(n) \leq c_2^n$$

для некоторых постоянных c_1 и c_2

4.2. k -счетчиковые и 3-счетчиковые машины.

Теорема 4.3. (а) Для данной k -СМ, работающей линейное время, можно найти эквивалентную 3-СМ, которая работает время $T(n) = cn^{k+2}$ для некоторой постоянной c . (б) Существует язык L , распознаваемый в реальное время k -СМ, который не распознается никакой r -СМ, работающей время меньше чем $n^{k/r}$.

Доказательство. (а) Если дана k -СМ M , работающая линейное время $d \cdot n$ (d — константа), можно легко построить эквивалентную 1-ТМ, которая (1) работает в течение времени $d' \cdot n \cdot \log n$ (для некоторого d'), (2) работает на пространстве $\log dn$ и (3) употребляет $2^k + 1$ рабочий символ. Эта 1-ТМ кодирует содержимое счетчиков M в двоичном алфавите на ленте, употребляя на каждый счетчик по одной дорожке. Изменение содержимого счетчиков M моделируется в виде соответствующего изменения двоичных чисел на соответствующих дорожках. Легко проверить, что если M изменяет не более одного счетчика на каждом шаге, то число шагов 1-ТМ, моделирующей проверку машиной M n -го входного символа, не превышает $\log dn$. Поэтому наша 1-ТМ удовлетворяет условиям (1) и (2).

Построенную таким образом 1-ТМ мы будем моделировать теперь при помощи 3-СМ, используя для этого алгоритм теоремы 3.1. Чтобы промоделировать один шаг работы 1-ТМ, который в свою очередь моделирует проверку машиной M n -го входного символа, 3-СМ должна умножить число, не большее $(2^k + 1)^{\log dn} \leq en^{k+1/2}$ (e — константа), на число $2^k + 1$, затем разделить число такого же порядка на $2^k + 1$ и прибавить величину, не большую $2^k + 1$. Каждый определенный таким образом шаг требует не более $3 \cdot (2^k + 1) \cdot e \cdot n^{k+1/2}$ шагов 3-СМ.

Так как 1-ТМ работает в течение времени $d' \cdot n \cdot \log n$, то моделирующая ее 3-СМ работает в течение времени $h \cdot n^{k+3/2} \cdot \log n \leq c \cdot n^{k+2}$, что и утверждалось.

(б) Используем семейство языков

$$L_k = \{0^{n_1} 10^{m_2} 1 \dots 0^{m_k} \beta_i 0^{m_i} \mid 1 \leq i \leq k,$$

любое $m_j \geq 1$ и любое $\beta_i \notin \{0, 1\}\}$.

Очевидно каждый ядык L_k распознаваем в реальное время некоторой k -СМ. Из доказательства теоремы 3.3(б) следует, что никакая r -СМ не может распознать L_k менее чем за $cn^{k/r}$ шагов (c — константа). Теорема доказана.

5. ЯЗЫКИ, РАСПОЗНАВАЕМЫЕ В РЕАЛЬНОЕ ВРЕМЯ

Пусть $\mathcal{C}_k$ означает класс языков, распознаваемых в реальное время k -СМ ($k = 0, 1, 2, \dots$), и пусть $\mathcal{C} = \bigcup_k \mathcal{C}_k$. В этом разделе мы исследуем некоторые свойства классов $\mathcal{C}_k$ и $\mathcal{C}$.

5.1. Счетчиковая иерархия. Большинство из отрицательных результатов этого раздела основывается на следующей важной лемме.

Мы говорим, что слова x и y n -эквивалентны относительно языка L , обозначая $x E_n y \pmod{L}$, если для всех слов z длины, не превышающей n , xz содержится в L тогда и только тогда, когда yz содержится в L .

Обозначим через $T(M)$ язык, распознаваемый СМ M .

Лемма 5.1. Пусть M есть k -СМ, работающая реальное время s q состояниями. Число классов эквивалентности $E_n \pmod{T(M)}$ не превышает $q \cdot (n+1)^k \leq cn^k$ для некоторой константы c .

Доказательство легко проводится с помощью рассмотрения числа различных конфигураций памяти M , которые могут быть отличены друг от друга за n шагов (которые « n -неэквивалентны»).

Из леммы 5.1 сразу же получаем следующую теорему.

Теорема 5.1. При всех k $\mathcal{C}_k$ в строгом смысле вложен в $\mathcal{C}_{k+1}$.

Доказательство. Рассмотрим семейство языков L_k из теорем 3.3(б) и 4.3(б). Если даны два различных слова $x = 0^{m_1}1 \dots 10^{m_{k+1}}$ и $y = 0^{r_1}1 \dots 10^{r_{k+1}}$, где каждое m_j и $r_j \leq h$, то существует последовательность $z = \beta_s 0^t$ длины, не превышающей $h+1$, такая, что xz содержится в L_{k+1} , а yz — нет. Поскольку число различных слов такого вида есть h^{k+1} , то число классов эквивалентности $E_{h+1} \pmod{L_{k+1}}$ не меньше чем h^{k+1} .

Теперь, поскольку число классов эквивалентности $E_{h+1} \pmod{T(M)}$ для любой k -СМ M , работающей реальное время, не больше, чем $q(h+1)^k < h^{k+1}$ для больших h , отсюда следует, что L_{k+1} не содержится в $\mathcal{C}_k$.

Очевидно, L_{k+1} содержится в $\mathcal{C}_{k+1}$, откуда для всех k $L_{k+1} \in \mathcal{C}_{k+1} \setminus \mathcal{C}_k$. Теорема доказана.

Лемма 5.1 и теорема 5.1 были независимо установлены Лэйнгом [7].

5.2. Линейное время и ускорение. Используя часть (б) теоремы 1.2, можно сразу же получить теорему «ускорения» для СМ.

Теорема 5.2. Для любой k -СМ, работающей время $T(n) = n + E(n)$, $E(n) \geq 0$, и для любой константы $c > 0$ можно эффективно найти эквивалентную k -СМ, которая работает время $T'(n) = n + [cE(n)]$.

В частности, получаем такое следствие.

Следствие. Для любой k -СМ, которая работает время $T(n) = d \cdot n$, $d \geq 1$, и для любого $c > 0$ можно найти эквивалентную k -СМ, которая работает время $(1 + [(d - 1)/c]) \cdot n$.

Итак, любая СМ, работающая линейное время, может быть заменена эквивалентной СМ, работающей время $(1 + \varepsilon) \cdot n$. Естественно спросить, можно ли положить здесь $\varepsilon = 0$. Мы покажем, что, вообще говоря, этого сделать нельзя.

Теорема 5.3 Пусть $L = \{0^p 1^m \mid p \geq m > 0\}$. Язык L^* (а) распознаем во время $2n$ с помощью 1-СМ, но (б) не распознаваем в реальное время никакой СМ.

Доказательство. (а) очевидно; предположив противное, приведем (б) к противоречию. Предположим, что k -СМ M с q состояниями распознает L^* в реальное время.

Пусть $b_0 = q$, и для $i = 1, \dots, k$ пусть $b_i = q \cdot (3b_{i-1} + 2)^i$.

Мы покажем, что M допускает слово вида $0^{c_k} 10^{c_{k-1}} \dots \dots 0^{c_1} 10^d 1^e$, если $d < e$, или отвергает это слово, если $d \geq e$. Числа c_i будут удовлетворять неравенствам $1 \leq c_i \leq b_i$, $i = 1, \dots, k$.

Теперь для любых целых чисел p и $m > p$ M должна иметь различные конфигурации памяти после чтения 0^p и 0^m , иначе слова $0^m 1^m \in L^*$ и $0^p 1^m \notin L^*$ будут восприняты одинаково. Поскольку число различных конфигураций памяти ограничено величиной $q \cdot (t + 1)^k$, где t — наибольшее из чисел, содержащихся в счетчиках, то после чтения слова 0^{c_k} , $c_k \leq b_k$, некоторый счетчик M должен содержать $t \geq 3b_{k-1} + 1$. Следовательно, после прочтения $0^{c_k} 1$ этот счетчик M содержит число не меньшее чем $3b_{k-1}$.

Пусть в качестве предположения индукции справедливо, что для $0 < i < k$ числа $c_k, \dots, c_{k-i+1}$ выбраны так, что когда M считывает последний символ слова

$$0^{c_k} 10^{c_{k-1}} \dots 10^{c_{k-i+1}} 1,$$

по крайней мере i счетчиков M содержат числа, не меньшие чем $3b_{k-i}$. Как и выше, после просмотра машиной M входных слов вида 0^p и 0^m для $b_{k-i} \geq m > p \geq 1$ конфигурации ее памяти должны быть b_{k-i} -неэквивалентными. По предположению индукции i счетчиков M содержат слишком большие числа, чтобы влиять на b_{k-i} -эквивалентность конфигураций. Следовательно, M теперь может иметь не более $q \cdot (t+1)^{k-i}$ b_{k-i} -неэквивалентных конфигураций, где t — наибольшее из чисел, содержащихся в $k-i$ «влияющих на отличимость» счетчиках. Значит, существует целое $c_{k-i} \leq b_{k-i}$, такое, что после прочтения слова

$$0^{c_k} 1 \dots 0^{c_{k-i+1}} 1 0^{c_{k-i}} 1$$

по крайней мере $i+1$ из счетчиков M содержат числа, не меньшие $3b_{k-i-1}$.

По предположению индукции должно быть такое входное слово $w = 0^{c_k} 1 \dots 0^{c_1} 1$ (все $c_i \leq b_i$), что после прочтения w все счетчики M содержат числа, не меньшие чем $3b_0 = 3q$. Но тогда должны быть числа c и d , $1 \leq c < d \leq q+1$, такие, что $w0^c$ и $w0^d$ приводят M в q -эквивалентные конфигурации. Следовательно, M должна одинаково воспринимать слова

$$w0^{c-1} 1^{+1} \notin L^* \quad \text{и} \quad w0^d 1^{c+1} \in L^*,$$

что противоречит условию о распознаваемости L^* машиной M . Доказательство окончено.

Теорема 5.3, таким образом, показывает неэквивалентность линейного и реального времени для СМ. Следует отметить, что L^* распознается в реальное время 1-СМ, которая может за один шаг устанавливать в счетчиках нуль. Теорема 5.3 дает следующее

Замечание. Класс языков, распознаваемых в реальное время СМ с командами «установки нуля», собственно содержит $\mathcal{E}$.

Читатель легко проверит, что почти все результаты данной работы остаются справедливыми для этих усиленных СМ.

5.3. Свойства замкнутости. В этом заключительном разделе мы исследуем свойства замкнутости классов $\mathcal{E}_k$ и $\mathcal{E}$. Свойства замкнутости класса $\mathcal{E}$ почти идентичны свойствам класса языков, определенных в реальное время (Розенберг [13]).

Теорема 5.4. Класс $\mathcal{E}$ замкнут относительно операции (а) дополнения, (б) объединения, (в) пересечения, (г) умножения справа на регулярное множество, (д) взятия прообраза при отображении обобщенной последовательностной машиной¹⁾.

¹⁾ См. статью [15*]. — Прим. ред.

Конструкции для доказательства можно заимствовать из работы Розенберга [13], в которой аналогичные результаты устанавливаются для машины Тьюринга.

Лемма 5.2. *В $\mathcal{C}$ не содержится язык*

$$L = \{0^{m_1}1 \dots 0^{m_{r-1}}10^{m_r}20^m \mid r \geq 1, \text{ все } m_j \geq 1, \\ \text{при некотором } i, 1 \leq i \leq r, m = m_i\}.$$

Доказательство. Если даны двоичные слова вида $x = 0^{m_1}1 \dots 0^{m_r}1$ ($r \geq 1, 1 \leq m_j \leq h$) и $y = 0^{n_1}1 \dots 0^{n_s}1$ ($s \geq 1, 1 \leq n_j \leq h$), то при $\{m_1, \dots, m_r\} \neq \{n_1, \dots, n_s\}$ существует слово $z \in \{2\} \cdot \{0\}^*$ длины, не большей $h+1$, такое, что $xz \in L$, а $yz \notin L$. Отсюда следует, что число классов эквивалентности $E_n \pmod{L}$ не меньше чем 2^{n-1} . Результат теперь следует из леммы 5.1.

Теорема 5.5. *Класс $\mathcal{C}$ не замкнут относительно операций (а) конкатенации даже с регулярным множеством, (б) гомоморфизма, сохраняющего длину, (в) итерации и (г) обращения.*

Доказательство. (а) — (в). Пусть

$$H = \{0^m 10^{n_1}1 \dots 0^{n_r}20^m \mid r \geq 0, \text{ каждое } n_i \geq 1 \text{ и } m \geq 1\}.$$

Ясно, что $H \in \mathcal{C}_1$. Пусть R — регулярное множество, $R = (\{0\}\{0\}^*\{1\})^*$. Поскольку $RH = L$, часть (а) следует из леммы 5.2. Пусть f есть сохраняющий длину гомоморфизм, определяемый так:

$$f(0) = 0, \quad f(1) = 1, \quad f(2) = 2, \quad f(3) = 1.$$

Ясно, что $R\{0\}\{0\}^*\{3\}H \in \mathcal{C}_1$. Однако образ этого языка относительно f есть L , что доказывает (б).

По теореме 5.4 $G = H\{3\} \cup R \in \mathcal{C}$. Однако $G^* \notin \mathcal{C}$, иначе, согласно теореме 5.4, было бы $G^* \cap R\{2\} \cdot \{0\}^* \cdot \{3\} = L \cdot \{3\}$, что противоречит лемме 5.2. Таким образом, установлено (в).

Легко проверить, что язык $K = \{1^n 0^m \mid 0 < n \leq m\}^*$ принадлежит $\mathcal{C}_1$. Однако обращение K есть язык L^* из теоремы 5.3, о котором мы знаем, что он не принадлежит $\mathcal{C}$.

Наконец, отметим еще одну теорему.

Теорема 5.6. *Для всех $m, n > 0$ существуют язык $H \in \mathcal{C}_m$ и язык $K \in \mathcal{C}_n$, такие, что $H \cup K \in \mathcal{C}_{m+n} \setminus \mathcal{C}_{m+n-1}$.*

Доказательство. Положим

$$H = \{0^{a_1}1 \dots 0^{a_m}10^{b_1}1 \dots 0^{b_n}\beta_i 0^{a_i} \mid 1 \leq i \leq m, \\ \text{все } a_i, b_j \geq 1, \text{ все } \beta_k \notin \{0, 1\}\}, \\ K = \{0^{c_1}1 \dots 0^{c_m}10^{d_1}1 \dots 0^{d_n}\beta_i 0^{d_i} \mid 1 \leq i \leq n, \\ \text{все } c_i, d_i \geq 1, \text{ все } \beta_k \notin \{0, 1\}\}.$$

Очевидно, NUK есть язык L_{m+n} теоремы 5.1, и, следовательно, он принадлежит $\mathcal{C}_{m+n} \setminus \mathcal{C}_{m+n-1}$.

Открытая проблема. Справедлив ли аналог теоремы 5.6 для ТМ?

ЛИТЕРАТУРА

- 1 Cobham A., Time and memory capacity bounds for machines, which recognize squares or palindromes, IBM Research Report RC-1621, 1966.
- 2 Fisher P. C., Turing machines with restricted memory access, *Information and Control*, 9 (1966), 364—379.
- 3 Fisher M. J., Rosenberg A. L., Real-time solutions of the origin-crossing problem, *Math. Systems Theory*, 2 (1968), 257—263. (См. стр. 372—379 настоящего сборника.)
- 4 Ginsburg S., The mathematical theory of context-free languages, McGraw-Hill, New York, 1966. (Русский перевод: Гинзбург С., Математическая теория контекстно-свободных языков, «Мир», М., 1970.)
- 5 Ginsburg S., Spanier E. H., Bounded ALGOL-like languages, *Trans. Amer. Math. Soc.*, 113 (1964), 333—368.
- 6 Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, 117 (1965), 285—306 (Русский перевод: Хартманис Дж., Стернс Р. Е., О вычислительной сложности алгоритмов, Кибернетический сборник (новая серия), вып. 4, 1967.)
- 7 Laing R., Realization and complexity of commutative events, Univ. of Mich. Technical Report 39105-48-T, 1967.
- 8 Minsky M., Recursive unsolvability of Post's problem of tag and other topics in the theory of Turing machines, *Ann. of Math.*, 74 (1961), 437—455.
- 9 Oettinger A. G., Automatic syntactic analysis and the pushdown store, Proc. 12th Ann. Sympos. in Appl. Math., 1960, 104—129.
- 10 Parikh R. J., On context-free languages, *J. Assoc. Comput. Mach.*, 13 (1966), 570—581.
- 11 Rabin M. O., Real-time computation, *Israel J. Math.*, 1 (1963), 203—211. (См. стр. 156—157 настоящего сборника.)
- 12 Rabin M. O., Scott D., Finite automata and their decision problems, *IBM J. Res. Develop.*, 3 (1959), 114—125. (Русский перевод: Рэбин М. О., Скотт Д., Конечные автоматы и проблемы их разрешения, Кибернетический сборник, вып. 4, ИЛ, М., 1962.)
- 13 Rosenberg A. L., Real-time definable languages. *J. Assoc. Comput. Mach.*, 14 (1967), 645—662. (См. стр. 168—193 настоящего сборника.)
- 14 Stearns R. E., Hartmanis J., Lewis P., Hierarchies of memory-limited computations, Proc. 6th Ann. Sympos. Switching Circuit Theory and Logical Design, Ann Arbor, Mich., 1966, 179—190. (См. стр. 301—319 настоящего сборника.)
- 15* Ginsburg S., Rose G., A characterization of machine mappings, *Canad. J. Math.*, 18, № 2, 381—388. (Русский перевод: Кибернетический сборник (новая серия), вып. 5, 1968, 128—137.)

МАШИННО-НЕЗАВИСИМАЯ ТЕОРИЯ СЛОЖНОСТИ РЕКУРСИВНЫХ ФУНКЦИЙ ¹⁾

М. Блюм

Число шагов, требуемых для вычисления значения функции, зависит, вообще говоря, от типа используемого вычислительного устройства, выбора программы вычислений и кодировки входных и выходных данных. Тем не менее результаты, полученные в этой работе, являются настолько общими, что они почти не зависят от этих условий.

Строится функция, которая требует для своего вычисления огромного числа шагов, однако она имеет почти «быстрейшую» программу: любая другая программа для вычисления этой функции, как бы она изобретательно ни была построена, делает практически столько же шагов, как и эта почти быстрейшая программа.

Другая построенная функция обладает тем свойством, что, как бы быстро ни работала программа для вычисления этой функции, существует другая программа, вычисляющая ту же функцию и работающая значительно быстрее.

1. ВВЕДЕНИЕ

Задача заключается в том, чтобы охарактеризовать сложность вычислимых функций. Развиваемая здесь теория следует духу предложенного Рабином аксиоматического подхода [5]. Выбранные аксиомы служат основой для определения того, что является, а что не является законной мерой функциональной сложности. Вероятно, наиболее известны такие интуитивные меры, как 1) число шагов, необходимых для вычисления функции, 2) размер ленты, необходимый машине для вычисления. Обе эти меры удовлетворяют нашим аксиомам. После формулировки аксиом будут даны и другие примеры.

Теория сложности, предлагаемая здесь, машинно-независима. Это означает, что теорема, которая характеризует сложности частично рекурсивных функций на одном классе машин, одинаково хорошо характеризует их сложности на почти любом другом классе. Хотя это и не является неожиданным, все же это утверждение представляется странным, так как сложность конкретных функций неизбежно зависит от класса машин, используемых для вычислений. Поэтому нужно меньшее число шагов

¹⁾ Blum M., A machine-independent theory of the complexity of recursive functions, *Journal of the Association for Computing Machinery*, 14, № 2 (1967), 322—336.

при вычислении функции в классе многоленточных машин по сравнению с классом одноленточных машин; в классе машин с двоичной кодировкой входных и выходных данных можно вычислять некоторые функции (такие, как 2^n) в меньшее число шагов, чем в классе машин с десятичной кодировкой. Таким образом, не должно быть надежды на то, что индивидуальная функция может иметь единственную меру сложности, которая не зависит от класса машин. Остается возможность, которая и предлагается здесь вместо этого: аксиоматическая теория, теоремы которой не зависят от класса. Пример такой теоремы поможет сформулировать этот подход.

Теорема 1. *Для любой общерекурсивной функции g можно построить общерекурсивную функцию f , принимающую значения 0 и 1 и имеющую такую сложность, что любая машина, которая вычисляет $f(n)$, делает больше чем $g(n)$ шагов для бесконечно многих значений n .*

Хотя сначала должны были бы быть введены аксиомы, они настолько очевидны, что неформальное доказательство этой теоремы может быть дано уже сейчас. В этом доказательстве $Z_0, Z_1, Z_2, \dots$ — последовательность машин, которые вычисляют все частично рекурсивные функции на натуральных числах.

Доказательство. Дана функция g , а функция f должна быть определена. В нашем случае она определяется с помощью диагонального метода. Для вычисления $f(n)$ имитируется n -я машина Z_n на входе n . Если Z_n останавливается менее чем за $g(n)$ шагов с некоторым выходом m , полагаем $f(n) \neq m$. В противном случае $f(n)$ может быть определена произвольно. Тогда любая машина Z_i , которая делает на входе i меньше чем $g(i)$ шагов, не может вычислять f . Но каждая машина встречается в приведенном выше списке бесконечно много раз (или мы можем так изменить список, чтобы добиться этого), в частности бесконечно много раз встречается каждая машина для f . Отсюда следует теорема.

Эта первая теорема не предназначена для того, чтобы произвести впечатление. Цель здесь заключается в том, чтобы показать, что существует теорема, которая истинна для любого класса машин независимо от того, как эти машины устроены и как у них кодируются данные. Так, $Z_0, Z_1, Z_2, \dots$ может быть классом одноленточных машин так же, как и классом 10-ленточных машин. Машина может быть даже такова, что входные данные кодируются в единичном коде, а выходные — с помощью римских цифр.

Мы свяжем с каждой машиной Z_i две функции: 1) частично рекурсивную функцию $\varphi_i(n)$, которая на этой машине вычис-

ляется, и 2) частично рекурсивную функцию $\Phi_i(n)$, называемую сигнализирующей функцией. В частности, $\Phi_i(n)$ можно понимать как число шагов или размер ленты, используемой Z_i , когда она работает на входе n .

В этой статье вводится ряд аксиом и доказывается несколько промежуточных теорем и две главные теоремы: теоремы ускорения (speed-up) и сжатия (compression). Теоремы ускорения и сжатия касаются следующей проблемы: дана рекурсивная функция f ; каждой ли машине, вычисляющей ее, соответствует другая машина, работающая быстрее?

Теорема ускорения или, вернее, ее частный случай дает примеры общерекурсивных функций f , принимающих два значения 0 и 1 и обладающих тем свойством, что для каждой Z_i , которая вычисляет f (за $\Phi_i(n)$ шагов), существует другая машина Z_j , которая делает ту же работу значительно быстрее (за $\Phi_j(n)$ шагов), причем $\Phi_i(n) > 2^{\Phi_j(n)}$ для почти всех n , т. е. для всех, за исключением конечного множества чисел. Отметим, что это приводит к бесконечной последовательности $f = \varphi_i = \varphi_j = \varphi_k = \dots$, такой, что

$$\Phi_i(n) > 2^{\Phi_j(n)} > 2^{2^{\Phi_k(n)}} > \dots$$

Теорема сжатия представляет собой обращение теоремы ускорения. Она весьма напоминает теорему 9 в работе Хартманиса и Стирнза [2], где показано, что у некоторых весьма сложных функций f число шагов, необходимых для их вычисления, можно «сжать» между двумя близкими границами. В этой статье доказана теорема совершенно общего характера, которая для случая одноленточных машин с алфавитом, состоящим из 0, 1 и b (пусто), имеющих двоичную кодировку входных и выходных данных, при функции $\Phi_i(n)$, определенной как действительное число шагов при вычислении $\varphi_i(n)$, дает следующее. Пусть φ_i — частично рекурсивная функция. Тогда существует принимающая значения 0 и 1 функция f , имеющая ту же область определения, что и φ_i , и имеющая такую сложность, что любая машина, вычисляющая $f(n)$, делает по крайней мере $\Phi_i(n)$ шагов для почти всех n , но при этом по крайней мере одна машина вычисляет $f(n)$ менее чем за $[\Phi_i(n)]^7$ шагов. Таким образом, число шагов, необходимых для вычисления f , заключено между Φ_i и Φ_i^7 .

2. АКСИОМЫ ДЛЯ СИГНАЛИЗИРУЮЩИХ ФУНКЦИЙ

Эти аксиомы могут быть сформулированы без ссылок на устройства или машины. Первый шаг заключается в постулировании эффективного списка $\{\varphi_i\}$ всех частично рекурсивных

функций и требования, чтобы для этого списка выполнялись S_n^m -теорема¹⁾ и теорема об универсальной машине Тьюринга²⁾. (Роджерс [8] называет любой такой список допустимой гёделевской нумерацией; он доказывает, что любые два таких списка рекурсивно изоморфны!) Второй шаг заключается в том, что с каждой φ_i связывается некоторая частично рекурсивная функция Φ_i . Множество $\{\Phi_i\}$ совершенно произвольно, на него накладываются лишь два основных ограничения, сформулированных в аксиомах:

1. $\varphi_i(n)$ определена (т. е. Z_i на входе n останавливается) $\Leftrightarrow \Phi_i(n)$ определена.

2. Функция

$$M(i, n, m) = \begin{cases} 1, & \text{если } \Phi_i(n) = m, \\ 0 & \text{в противном случае} \end{cases}$$

является (обще)рекурсивной.

Аксиому 1 можно выразить эквивалентно в терминах функции M так: $\varphi_i(n)$ определена $\Leftrightarrow \exists m[M(i, n, m) = 1]$. Функция M , которая удовлетворяет аксиомам 1 и 2, называется мерой вычисления.

В дальнейшем будем пользоваться сокращением $\Phi_i(n) > t$ для $\infty \geq \Phi_i(n) > t$, где $\Phi_i(n) = \infty$ означает, что $\varphi_i(n)$ не определена.

Пример 1. Пусть $\{Z_i\}$ — класс многоленточных машин, а $\{\varphi_i\}$ — множество вычислимых на них функций; тогда одно из возможных множеств сигнализирующих функций $\{\Phi_i\}$ определяется так: $\Phi_i(n) = t$ тогда и только тогда, когда Z_i на входе n останавливается точно за t шагов (в противном случае $\Phi_i(n)$ не определена).

Другое возможное множество $\{\Phi_i\}$ определяется так: $\Phi_i(n) = t$ тогда и только тогда, когда Z_i при работе на входе n использует точно t ячеек ленты.

Пример 2. Если $\{\Phi_i\}$ удовлетворяет аксиомам, то также удовлетворяет аксиомам $\{\hat{\Phi}_i\}$, определенное так: $\hat{\Phi}_i(n) = \Phi_i(n) \cdot 2^{i+n}$.

¹⁾ S_n^m -теорема [9], известная также как теорема об итерации, утверждает существование общерекурсивной функции σ , такой, что $\varphi_{\sigma(i, m)}(n) = \varphi_i(m, n)$ для всех i, n и m .

²⁾ Теорема об универсальной машине Тьюринга гласит, что для любого эффективного взаимно однозначного отображения $\tau: N \times N \rightarrow N$, где N — множество натуральных чисел, существует универсальная машина Z_i , обладающая свойством $\varphi_i(\tau(x, y)) = \varphi_x(y)$ для всех x и y .

Другой возможный выбор $\{\hat{\Phi}_i\}$, основывающийся на некоторой общерекурсивной функции φ_{i_0} , таков:

$$\hat{\Phi}_i(n) = \begin{cases} \Phi_i(n), & \text{если } i \neq i_0, \\ 0, & \text{если } i = i_0. \end{cases}$$

Пример 3. Выбор $\Phi_i(n) = \varphi_i(n)$ невозможен, так как выполняется лишь аксиома 1, аксиома 2 не выполняется; невозможен также выбор множества $\Phi_i(n) = 0$ для всех допустимых i и n , так как в этом случае выполняется аксиома 2, но не выполняется аксиома 1. Отсюда следует, что приведенные две аксиомы независимы.

Каждая сигнализирующая функция Φ_i , будучи частично рекурсивной, встречается где-то в списке $\{\varphi_i\}$. Вторая теорема утверждает, что существует эффективная процедура для определения этого места.

Теорема 2. *Существует общерекурсивная функция β , такая, что $\Phi_i = \varphi_{\beta(i)}$ для всех i .*

Доказательство. Аксиома 2 утверждает, что

$$M(i, n, m) = 1 \Leftrightarrow \Phi_i(n) = m.$$

Определим функцию f , которая удовлетворяет условию

$$f(i, n) = m \Leftrightarrow M(i, n, m) = 1.$$

Функция f частично рекурсивна. Согласно S_n^m -теореме существует общерекурсивная функция β , такая, что $\varphi_{\beta(i)}(n) = f(i, n)$. Функция β удовлетворяет требованиям теоремы.

Третья теорема уточняет следующее утверждение. Пусть даны два класса машин и вычислимая функция f . Тогда для каждой машины из одного класса, вычисляющей f , существует машина из другого класса, которая вычисляет функцию f примерно за то же число шагов.

Теорема 3. *Пусть M и $\hat{M}$ — произвольные меры вычислений с сигнализирующими функциями $\{\Phi_i\}$ и $\{\hat{\Phi}_i\}$. Тогда существует такая общерекурсивная функция g , что*

$$g(n, \hat{\Phi}_i(n)) \geq \Phi_i(n) \quad \text{и} \quad g(n, \Phi_i(n)) \geq \hat{\Phi}_i(n)$$

для всех i и почти всех n .

Это означает, что $\Phi_i(n)$ и $\hat{\Phi}_i(n)$ не слишком отличаются друг от друга.

Доказательство. Требуемая функция g получается из функции p , определяемой так:

$$p(i, n, m) = \begin{cases} \Phi_i(n) + \hat{\Phi}_i(n), & \text{если } \Phi_i(n) = m \text{ или } \hat{\Phi}_i(n) = m, \\ \text{т. е. если } M(i, n, m) = 1 \text{ или } \hat{M}(i, n, m) = 1, \\ 0 & \text{в противном случае.} \end{cases}$$

Функция p общерекурсивна; она вычислима, так как вычислимы M и $\hat{M}$, и всюду определена, так как $(\hat{\Phi}_i(n) \text{ определена}) \Leftrightarrow (\varphi_i(n) \text{ определена}) \Leftrightarrow (\Phi_i(n) \text{ определена})$. Требуемая функция g задается так: $g(n, m) = \max_{i \leq n} p(i, n, m)$, что и требовалось доказать.

Удобным средством для доказательства теоремы о сложности является использование упрощенного варианта теоремы о рекурсии Клини [9]: для каждой общерекурсивной функции σ найдется такое натуральное число j , что $\varphi_{\sigma(j)} = \varphi_j$. Примером использования этой теоремы может служить доказательство очевидного утверждения: плохо построенная машина может тратить зря огромное число шагов при вычислении простой функции.

Теорема 4. Пусть h и f — общерекурсивные функции. Тогда существует такой индекс j для f , что $\Phi_j(n) > h(n)$ для всех n .

Доказательство. Пусть

$$p(i, n) = \begin{cases} f(n), & \text{если } \Phi_i(n) > h(n), \text{ т. е. если } M(i, n, m) = 0 \\ & \text{для всех } m \leq h(n), \\ 1 + \varphi_i(n) & \text{в противном случае.} \end{cases}$$

Функция p общерекурсивна, поэтому существует общерекурсивная функция σ , такая, что $p(i, n) = \varphi_{\sigma(i)}(n)$. Согласно теореме о рекурсии, найдется такое число j , что $\varphi_{\sigma(j)} = \varphi_j$. Теперь легко видеть, что j — индекс для f : если $\Phi_j(n) \leq h(n)$, то $\varphi_j(n) = 1 + \varphi_j(n)$, что противоречиво. Поэтому $\Phi_j(n) > h(n)$ для всех n . Из определения функции p следует, что $\varphi_j(n) = f(n)$ для всех n . Что и требовалось доказать.

3. УСКОРЕНИЕ

На одноленточной машине с десятичной кодировкой входных и выходных данных при сигнализирующей функции, пересчитывающей действительное число шагов, функции, подобные $f(n) = 0$ или $f(n) = n$, могут быть вычислены быстрее или наилучшим способом. Это неудивительно, поскольку эти функции вычислимы на конечных автоматах.

С другой стороны, если функция достаточно сложна, то для каждой машины, которая ее вычисляет, есть другая машина, более быстрая, которая вычисляет ее за точно вдвое меньшее число шагов для бесконечно многих n . Простой пример такой функции:

$$f(n) = \begin{cases} 1, & \text{если } n \text{ — палиндромическое число}^1), \\ 0 & \text{в противном случае.} \end{cases}$$

Эта функция определяет, читается ли число, записанное в десятичном коде, слева направо так же, как справа налево. Чтобы вычислить $f(n)$, $n=372686273$, обычная машина сначала читает самую правую цифру 3, стирает ее, затем бежит по ленте к самой левой цифре 3 и стирает ее, возвращается обратно на самую правую цифру 7, стирает ее, бежит к самой левой цифре 7 и стирает ее и т. д. После сравнения соседних цифр она печатает 1. Более быстрая машина читает сразу самые правые цифры 3 и 7 перед тем, как двигаться к самым левым цифрам 3 и 7, при этом она сравнивает сразу две цифры, в то время как первая машина за один прием сравнивает одну цифру. Эта более быстрая машина делает примерно в половину меньшее число шагов, чем более медленная, для всех палиндромических чисел. На самом деле можно доказать (хотя и не просто), что, какая бы ни была выбрана машина для вычисления этой функции, может быть построена другая машина, которая делает ту же самую работу за точно вдвое меньшее число шагов для бесконечно многих входов.

Это показывает, что верен следующий результат: существует общерекурсивная функция f , обладающая тем свойством, что каждому индексу i для этой функции соответствует другой индекс j для f , такой, что $\Phi_i(n) > 2\Phi_j(n)$ для почти всех n . Может ли быть доказана более сильная теорема? В частности, существует ли функция f , обладающая свойством, что каждому индексу i для этой функции соответствует индексу j для f , такой, что

$$\Phi_i(n) > \Phi_j(n)^{\Phi_i(n)}$$

для почти всех n ? Следующая теорема утверждает, что такая функция существует.

Теорема 5 (ускорения). Пусть r — общерекурсивная функция от двух переменных. Тогда существует общерекурсивная функция f , принимающая значения 0 и 1, с тем свойством, что каждому индексу i для f соответствует другой индекс j для f , такой, что $\Phi_i(n) > r(n, \Phi_j(n))$ для почти всех n .

¹⁾ Палиндромическое число — число с симметрично расположенными цифрами. — Прим перев.

Доказательство. В основе этой теоремы лежат простые соображения. Они следуют из прямого доказательства того, что каждой общерекурсивной h соответствует общерекурсивная f со свойством, что если i — любой индекс для f , то $\Phi_i(n) > h(n)$ для почти всех n . Это прямое доказательство определяет f , задавая ее вычисление с помощью вычеркивания первой невычеркнутой функции $\varphi_i(n)$, которая принадлежит множеству $\{\varphi_0(n), \dots, \varphi_n(n)\}$ и такова, что ее сигнализирующая функция Φ_i обладает свойством $\Phi_i(n) \leq h(n)$. Далее полагается $f(n) \neq \varphi_i(n)$; чтобы избежать неприятности, связанной с тем, что нельзя рассматривать i -ю функцию φ_i в последующих точках, вычеркнутая функция φ_i исключается из вычисления f для больших n .

Небольшое изменение в конструкции превращает f в функцию, удовлетворяющую теореме: вместо вычеркивания первого еще невычеркнутого элемента множества $\{\varphi_0(n), \dots, \varphi_n(n)\}$ со свойством $\Phi_i(n) \leq h(n)$ мы требуем вычеркивания первого элемента со свойством $\Phi_i(n) \leq h(n-i)$. Далее, как и раньше, полагаем $f(n) \neq \varphi_i(n)$. Так определяется f . В этом доказательстве вычисление задает функцию f , такую, что $f = \varphi_{l(0,0,1)}$, где l — индекс для h .

Интуитивные соображения опираются прежде всего на определение функции f . Машина, вычисляющая $\varphi_{l(0,0,1)}$, просматривает множество $\{\varphi_0(n), \dots, \varphi_n(n)\}$ и вычеркивает первую невычеркнутую φ_i , для которой $\Phi_i(n) \leq \varphi_l(n-1)$. Более быстрый способ вычисления $f(n)$ использует то обстоятельство, что на самом деле нет необходимости просматривать все множество $\{\varphi_0(n), \dots, \varphi_n(n)\}$. Для данного фиксированного u достаточно для больших n просмотреть только множество $\{\varphi_u(n), \dots, \varphi_n(n)\}$. Причина этого заключается в том, что для достаточно больших n каждый из элементов $\varphi_0(n), \dots, \varphi_{u-1}(n)$ или уже вычеркнут или никогда не будет вычеркнут, поэтому ни один из них не будет вычеркнут при вычислении $f(n)$. Эти элементы можно пропустить при достаточно больших n , скажем $n \geq v$. При так урезанном вычислении, при котором просматривается меньшее множество $\{\varphi_u(n), \dots, \varphi_n(n)\}$, делается меньшее число шагов, поскольку $f = \varphi_{l(u,v,1)}$. Вообще, чем больше u , тем быстрее работает соответствующая машина, которая вычисляет f . Этого возрастания достаточно, чтобы доказать теорему.

Возникает лишь небольшая проблема. Эти вычисления, использующие u и v , — особый способ вычисления f , и может показаться странным, что такой более быстрый способ вообще существует. Однако это именно так, так как если j — любой индекс для f , то вследствие определения f $\Phi_j(n) > \varphi_l(n-j)$ для почти всех n ; функция $\Phi_j(n)$ выбирается в качестве нижней границы числа шагов, которое выполняется при вычислении $f(n)$.

Эта более низкая граница используется, чтобы показать, что любой процесс вычисления f не лучше, чем описанный выше при фиксированных числах u и v .

Доказательство этой теоремы следует плану, предложенному М. Арбибом; здесь оно сделано более ясным и сокращено. Определяется критическая общерекурсивная функция $t(u, v, l)$, такая, что если φ_l всюду определена, то $\varphi_{t(u, v, l)}$ всюду определена и принимает значения 0 и 1. Затем доказываются:

Лемма 1. Если φ_l всюду определена, то для каждого u существует v , такое, что $\varphi_{t(u, v, l)} = \varphi_{t(0, 0, l)}$.

Лемма 2. Если φ_l всюду определена и если мы определим $f = \varphi_{t(0, 0, l)}$, то для каждого индекса i для f $\Phi_i(n) > \varphi_l(n - i)$ для почти всех n .

Лемма 3. Существует общерекурсивная функция φ_l , такая, что для всех u и v имеем $\varphi_l(n - u + 1) \geq r(n, \Phi_{t(u, v, l)}(n))$ для почти всех n .

Далее для каждого индекса i для $f = \varphi_{t(0, 0, l)}$ и для почти всех n

$$\begin{aligned} \Phi_i(n) &> \varphi_l(n - i) && \text{(лемма 2),} \\ &\geq r(n, \Phi_{t(i+1, v, l)}(n)) && \text{для всех } v \text{ (полагая } u = i + 1 \\ &&& \text{в лемме 3),} \\ &= r(n, \Phi_f(n)) && \text{(по лемме 1, выбирая } v \text{ таким, что} \\ &&& f = \varphi_{t(i+1, v, l)} = \varphi_f), \end{aligned}$$

отсюда следует теорема. Что и требовалось доказать.

Обратимся теперь к построению t и доказательству этих трех лемм.

1. Построение t . Функция $\varphi_{t(u, v, l)}$ определяется в терминах множества (рис. 1):

$$C_{uv} = \{\varphi_k(m) \mid (m < v \text{ и } k \leq m) \text{ или } (m \geq v \text{ и } u \leq k \leq m)\}.$$

Вычисляем $\varphi_{t(u, v, l)}(n)$ так:

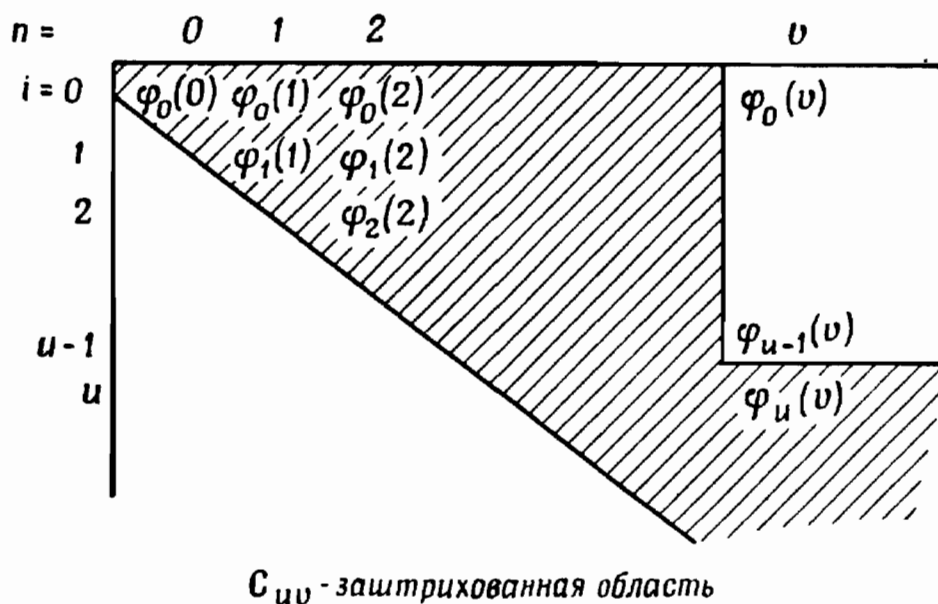
(1) если $v < u$, полагаем $\varphi_{t(u, v, l)}(n) = \varphi_{t(u, u, l)}(n)$ и используем (2).

(2) если $v \geq u$, поступаем так:

(α) вычисляем $\varphi_l(0), \dots, \varphi_l(n)$, если $n < v$, но при $n \geq v$ вычисляем только $\varphi_l(0), \dots, \varphi_l(n - u)$. Если все они не определены, считаем, что $\varphi_{t(u, v, l)}(n)$ не определена. В противном случае переходим к (β).

(β) Помечаем все $\varphi_k(n)$ в пересечении C_{uv} и n -го столбца, такие, что $\Phi_k(n) \leq \varphi_l(n - k)$. Если нам ничего не удалось пометить, полагаем $\varphi_{t(u, v, l)}(n)$ равной нулю. В противном случае переходим к (γ).

(γ) Вычеркнем первое вхождение $\varphi_k(n)$ в столбец n , такое, что $\varphi_k(n)$ помечена и обладает тем свойством, что все $\varphi_k(m)$, $m < n$, не вычеркнуты. Если нам не удалось ничего вычеркнуть в пересечении C_{uv} и n -го столбца, полагаем $\varphi_{t(u, v, l)}(n)$ равной нулю. В противном случае переходим к (δ).



Р и с. 1.

(δ) Если $\varphi_k(n)$ вычеркнута, полагаем

$$\varphi_{t(u, v, l)}(n) = \begin{cases} 0, & \text{если } \varphi_k(n) \neq 0, \\ 1, & \text{если } \varphi_k(n) = 0. \end{cases}$$

Из этого определения следует, что если φ_l определена всюду, то $\varphi_{t(u, v, l)}$ определена для всех u и v .

2. Доказательство леммы 1. В вычислении $\varphi_{t(0, 0, l)}$ число описанных выше вхождений в строку u , которые вычеркиваются, должно быть конечным, так как строка содержит самое большее одно вычеркнутое вхождение. Пусть столбец v располагается непосредственно справа от самого правого вычеркнутого описанного выше вхождения в строку u . Тогда из рисунка ясно, что $\varphi_{t(0, 0, l)}(n) = \varphi_{t(u, v, l)}(n)$ для всех n . Что и требовалось доказать.

3. Доказательство леммы 2. Пусть i — некоторый индекс для f ; доказывая от противного, предположим, что $\Phi_i(n) \leq \leq \varphi_l(n-1)$ для бесконечно многих $n = n_1, n_2, n_3, \dots$. Тогда, очевидно, в вычислении $\varphi_{t(0, 0, l)}(n)$ $\varphi_i(n_k)$ была вычеркнута для некоторого k , и поэтому $\varphi_{t(0, 0, l)}(n_k) \neq \varphi_i(n_k)$, откуда следует, что $f(n_k) \neq \varphi(n_k)$. Противоречие. Что и требовалось доказать.

4. Доказательство леммы 3. Выберем некоторое рекурсивное взаимно однозначное отображение натуральных чисел на множество всех m -ок ($m = 1, 2, 3, \dots$) натуральных чисел. Пусть

$\langle\langle a_0, \dots, a_n \rangle\rangle$ обозначает число, которое отображается в $(a_0, \dots, a_n)$. Определим

$$g(n, u, v, i, z) = \begin{cases} \Phi_{i(u, v, i)}(n), & \text{если } n \geq u + v \text{ и} \\ & z = \langle\langle \Phi_i(0), \dots, \Phi_i(n - u) \rangle\rangle, \\ 0 & \text{в противном случае.} \end{cases}$$

Учитывая свойство M -функции, мы видим, что g общерекурсивна, поэтому достаточно найти такую общерекурсивную функцию φ_l , что для почти всех n

$$\varphi_l(n - u + 1) \geq r[n, g(n, u, v, l, \langle\langle \Phi_l(0), \dots, \Phi_l(n - u) \rangle\rangle)]. \quad (3.1)$$

Определим p так:

$$\begin{aligned} p(i, 0) &= 0, \\ p(i, z + 1) &= \max_{\substack{0 \leq v \leq z \\ 0 \leq u \leq z}} r[z + u, g(z + u, u, v, i, \langle\langle \Phi_i(0), \dots, \Phi_i(z) \rangle\rangle)]. \end{aligned} \quad (3.2)$$

Функция p частично рекурсивна, поэтому существует общерекурсивная функция σ , такая, что $p(i, z) = \varphi_{\sigma(i)}(z)$. Согласно теореме о рекурсии, существует индекс l , такой, что $\varphi_{\sigma(l)}(z) = \varphi_l(z)$ для всех z . Поэтому

$$\varphi_l(z) = p(l, z). \quad (3.3)$$

Таким образом, $\varphi_l(0) = p(l, 0) = 0$. Теперь допустим, что $\varphi_l(n)$ определена для $n \leq z$. Рассматривая (3.2) и вспоминая (3.3), мы видим, что $\varphi_l(z + 1)$ определена, так как g и r общерекурсивны, а из определенности $\varphi_l(n)$ для $n \leq z$ следует определенность $\Phi_l(n)$ для $n \leq z$. Поэтому φ_l общерекурсивна.

Фиксируем u и v . Тогда для $n \geq \max[2u, u + v]$ имеет место $v \leq n - u$, $u \leq n - u$, и поэтому, полагая в (3.2) $z = n - u$ и $i = l$, мы получаем (3.1). Что и требовалось доказать.

Для примера возьмем $r(n, m) = 2^m$. Тогда теорема ускорения утверждает существование такой общерекурсивной функции f , что любому индексу i для f соответствует такой другой индекс j для f , что $\Phi_i(n) > 2^{\Phi_j(n)}$ для почти всех n . Поэтому она также утверждает, что индексу j для f соответствует такой индекс k для f , что $\Phi_i(n) > 2^{2^{\Phi_k(n)}}$, и т. д. К сожалению, теорема ускорения не дает эффективной процедуры для перехода от машины, вычисляющей f , к машине, делающей это более быстро. Такой эффективной процедуры и не может быть, как это показывает следствие 1.

Следствие 1. Пусть r — некоторая общерекурсивная функция, а f — соответствующая функция из теоремы ускорения.

Если r достаточно велика, то не существует общерекурсивной функции κ , такой, что (1) κ перечисляет только индексы f , (2) каждому индексу i для f соответствует такой индекс $\kappa(j)$ для f , что $\Phi_i(n) > r(n, \Phi_{\kappa(j)}(n))$ для почти всех n .

Доказательство. Пусть π — общерекурсивная функция, обладающая тем свойством, что для каждого j существует бесконечно много n , таких, что $\pi(n) = j$. По S_n^m -теореме существует общерекурсивная функция σ , такая, что

$$\Phi_{\sigma(i)}(n) = \Phi_{\Phi_i(\pi(n))}(n) \quad (3.4)$$

для всех i и n . Для этой σ , как легко видеть, существует общерекурсивная функция g , такая, что

$$\Phi_{\sigma(i)}(n) = g(n, i, \Phi_i(\pi(n)), \Phi_{\Phi_i(\pi(n))}(n)),$$

и поэтому существует общерекурсивная функция h , такая, что

$$\Phi_{\sigma(i)}(n) < h(n, \Phi_{\Phi_i(\pi(n))}(n)) + h(i, \Phi_i(\pi(n))) \quad (3.5)$$

для всех i и n . Пусть $\kappa = \varphi_k$ перечисляет только индексы функции f . Тогда $\Phi_{\Phi_{\kappa}(\pi(n))}(n) = \Phi_{\kappa(\pi(n))}(n)$, и поэтому по (3.4) $\Phi_{\sigma(k)}(n) = \Phi_{\kappa(\pi(n))}(n)$; таким образом, $\sigma(k)$ есть индекс для f . По (3.5)

$$\Phi_{\sigma(k)}(n) < h(n, \Phi_{\kappa(\pi(n))}(n)) + h(k, \Phi_k(\pi(n))) \quad (3.6)$$

для всех n . Для каждого j и бесконечно многих n $\pi(n) = j$, и поэтому для каждого j и бесконечно многих n

$$\Phi_{\sigma(k)}(n) < h(n, \Phi_{\kappa(j)}(n)) + h(k, \Phi_k(j)). \quad (3.7)$$

Но h , согласно (3.5), не зависит от r . Поэтому, не изменяя h , можно выбрать r настолько большим, насколько это нужно. В частности, пусть r — любая функция, которая удовлетворяет неравенству $r(n, m) \geq n + h(n, m)$ для всех n и m . Если мы положим $m = \Phi_{\kappa(j)}(n)$, то (3.7) дает

$$\Phi_{\sigma(k)}(n) < r(n, \Phi_{\kappa(j)}(n)) - n + h(k, \Phi_k(j))$$

для бесконечно многих n . Отсюда в свою очередь следует, что для каждого j существует бесконечно много n , таких, что

$$\Phi_{\sigma(k)}(n) < r(n, \Phi_{\kappa(j)}(n)).$$

Как показано непосредственно перед (3.6), $f = \varphi_{\sigma(k)}$, поэтому получено противоречие с частью 2 этого следствия. Что и требовалось доказать.

В отношении r -ускорения $(\Phi_i(n) > r(n, \Phi_j(n)))$ можно предположить, что r должна быть чрезвычайно маленькой функцией по сравнению с Φ_j . Следующая теорема утверждает, что в действительности r может быть также велика, как и Φ_j .

Теорема 6 (сверхускорения). Пусть g — общерекурсивная функция. Тогда существует общерекурсивная функция f , принимающая значение 0 и 1 и такая, что (1) если i — индекс для f , то $\Phi_i(n) > g(n)$ для почти всех n ; (2) каждому индексу i для f соответствует такой индекс j для f , что $\Phi_i(n) > \Phi_j(\Phi_j(n))$ для почти всех n .

Доказательство. Доказательство этой теоремы сходно с доказательством теоремы ускорения, поэтому оно не приводится подробно, а только намечается. Доказательство основано на следующей лемме.

Лемма. Существует последовательность $\{p_s\}_{s=0}^{\infty}$ монотонно возрастающих общерекурсивных функций p_s , такая, что

- (1) для каждого s и почти всех n $p_s(n) > g(n)$;
- (2) для каждого s и почти всех n $p_s(n) > p_{s+1}(p_{s+1}(n))$;
- (3) для каждого s и всех n $p_s(n) \geq p_{s+1}(n)$.

Доказательство этой леммы предоставляется читателю.

Чтобы вычислить $f(n)$, вычеркнем первый невычеркнутый элемент φ_i , который принадлежит множеству $\{\varphi_0(n), \dots, \varphi_n(n)\}$ и сигнализирующая функция Φ_i которого обладает свойством $\Phi_i(n) \leq p_i(n)$. Затем полагаем $f(n) \neq \varphi_i(n)$. Отсюда следует, что если i — индекс для f , то для почти всех n

$$\Phi_i(n) > p_i(n). \quad (3.8)$$

Согласно лемме, $p_i(n) > g(n)$ для почти всех n ; таким образом, часть 1 теоремы доказана.

Чтобы вычислять $f(n)$ более быстро, достаточно зафиксировать число u и для достаточно больших n просматривать только множество $\{\varphi_u(n), \dots, \varphi_n(n)\}$. Может быть определено число шагов, необходимых для вычисления $f(n)$ указанным способом; может быть показано с помощью части 3 леммы, что существует общерекурсивная функция h , которая не зависит от последовательности $\{p_s\}$ и такая, что машина, вычисляющая $f(n)$ указанным путем (т. е. просматривая множество $\{\varphi_u(n), \dots, \varphi_n(n)\}$), делает меньше чем $h(n, p_u(n))$ шагов для почти всех n . Это означает, что для каждого числа u существует машина Z_j , которая вычисляет f так быстро, что для почти всех n

$$h(n, p_u(n)) > \Phi_j(n). \quad (3.9)$$

Так как функция h не зависит от выбора последовательности $\{p_s\}$, функция g может быть сделана в лемме настолько большой, что $p_{u-1}(n) > h(n, p_u(n))$ для почти всех n . Затем, если взять $u = i + 2$, это дает для почти всех n

$$p_{i+1}(n) > h(n, p_{i+2}(n)). \quad (3.10)$$

Отсюда следует, что если Z_i вычисляет f , то для почти всех n

$$\begin{aligned}
 \Phi_i(n) &> p_i(n) && \text{(по (3.8)),} \\
 &> p_{i+1}(p_{i+1}(n)) && \text{(лемма, часть 2),} \\
 &> p_{i+1}(h(n, p_{i+2}(n))) && \text{(по (3.10) и тому факту, что} \\
 & && p_{i+1} \text{ — монотонно возрастающая),} \\
 &> p_{i+1}(\Phi_j(n)), \text{ где } f = \varphi_j \text{ } (u = i + 2 \text{ в (3.9))}, \\
 &> h(\Phi_j(n), p_{i+2}(\Phi_j(n))) && \text{(по (3.10)),} \\
 &> \Phi_j(\Phi_j(n)), \text{ где } f = \varphi_j \text{ } (u = i + 2 \text{ в (3.9))},
 \end{aligned}$$

что и требовалось доказать.

4. СЖИМАНИЕ

Следующая теорема, которая сходна с теоремой Рабина [5], доказывает существование сколь угодно сложных функций, принимающих значение 0 и 1. Это напоминает теорему 1, но этот результат выходит за рамки теоремы 1 в трех важных направлениях. Если теорема 1 утверждает, что существуют функции $f(n)$, принимающие значения 0 и 1 и делающие больше чем $g(n)$ шагов при вычислении для бесконечно многих n , то следующая теорема утверждает то же самое для почти всех n . При этом от g и f требуется лишь частичная рекурсивность вместо общей рекурсивности. Наконец, доказываемая рекурсивность γ , которая отображает индексы функций g в индексы функций f . Теорема 7 не является тривиальным обобщением, и теорема сжатия существенно опирается на нее, особенно на факт существования γ .

Теорема 7. (1) *Каждой частично рекурсивной функции g соответствует частично рекурсивная функция f , принимающая значения 0, 1, с той же областью определения, что и g , и такая, что если j — любой индекс для f , то $\Phi_j(n) > g(n)$ для почти всех n .* (2) *Существует общерекурсивная функция γ , которая переводит индекс любой функции g в индекс для соответствующей функции f .*

Доказательство. Если заранее известно, что g общерекурсивна, то для вычисления f служит очень простая процедура. Для того чтобы вычислить $f(n)$, вычеркнем первый невычеркнутый элемент φ_j , который принадлежит множеству $\{\varphi_0(n), \dots, \varphi_n(n)\}$ и сигнализирующая функция которого обладает свойством $\Phi_j(n) \leq g(n)$. Затем полагаем $f(n) \neq \varphi_j(n)$. При

таким способом построения $f(n)$ делается отличной от тех функций φ_j , для которых $\Phi_j(n) \leq g(n)$; поэтому должно оказаться, что если k есть индекс для f , то $\Phi_k(n) > g(n)$ для почти всех n .

Эта процедура для вычисления $f(n)$ требует, чтобы было известно, какие функции в множестве $\{\varphi_0, \dots, \varphi_n\}$ были вычеркнуты в процессе вычисления $f(0), \dots, f(n-1)$. Поэтому для вычисления $f(n)$ по этой процедуре необходимо вначале вычислить $f(0)$, затем $f(1)$ и т. д. до $f(n-1)$. Это достаточно легко, если g всюду определена, но если это не так, $g(m)$ может быть не определена для некоторого m , а тогда также должна быть не определена $f(m)$, причем может не существовать способа выяснения, что $f(m)$ не определена. Если $m < n$, это делает невозможным вычисление $f(n)$. Чтобы обойти эту трудность, мы будем вычислять $f(n)$ так, что сначала не обязательно вычисляются значения от $f(0)$ до $f(n-1)$. Идея заключается в перечислении в определенном порядке всех пар неотрицательных целых чисел так, что для любых неотрицательных целых n и m пара (n, m) будет в конце концов получена.

После получения любой пары (n, m) возьмем индекс i для g и определим, имеет ли место $\Phi_i(n) = m$ или нет. Это может быть сделано эффективно, если учесть свойства функции меры вычислений M . Если $\Phi_i(n) \neq m$, переходим к следующей паре в списке. С другой стороны, если $\Phi_i(n) = m$, вычисляем $f(n)$ так: вычеркиваем первый элемент φ_j , принадлежащий множеству $\{\varphi_0, \dots, \varphi_n\}$, который не был вычеркнут при перечислении пар до данной пары (n, m) и сигнализирующая функция Φ_j которого обладает свойством $\Phi_j(n) \leq g(n)$. Возможно, ни один элемент φ_j не вычеркивается. Если это так, полагаем $f(n) = 1$. С другой стороны, если φ_j вычеркивается, полагаем $f(n) \neq \varphi_j(n)$. Это определяет f .

Доказательство следствия 1 теоремы 8 требует весьма подробного описания приведенной выше конструкции f . Это описание дается ниже и может быть при желании опущено при первом чтении.

Процедура перехода от индекса i для g и числа n к значению $f(n)$ такова:

Прежде всего вычисляем $g(n)$ ($g = \varphi_i$). Если $g(n)$ не определена, полагаем, что $f(n)$ не определена. Если $g(n)$ определена, $f(n)$ также должна быть определена. При вычислении $f(n)$ процедура исходит из того, что вычеркиваются все функции φ_j , сигнализирующие функции Φ_j которых обладают тем свойством, что $\Phi_j(n) \leq g(n)$ для бесконечно многих n , а соответствующие пары (j, n) присоединяются к списку L . Список L растет по шагам, причем на каждом шаге к L присоединяется не более одной пары (j, n) . Рост L на шаге x описан ниже. Рекурсивная

кодировка, с помощью которой $N \times N$ отображается на N , использована для того, чтобы переписать x в виде $\langle p, q \rangle = x$.

Шаг $x = \langle p, q \rangle$. Если $\Phi_i(p) \neq q$ ($g = \Phi_i$), то пропускаем шаг x и переходим к шагу $x+1$. С другой стороны, если $\Phi_i(p) = q$, то вычеркиваем первую невычеркнутую функцию $\Phi_j(p)$, если найдется некоторая функция, которая принадлежит множеству $\{\Phi_0(p), \dots, \Phi_p(p)\}$ и сигнализирующая функция которой такова, что $\Phi_j(p) \leq g(p)$. Вычеркнуть $\Phi_j(p)$ означает добавить (j, p) к списку L . После вычеркивания $\Phi_j(p)$ переходим к шагу $x+1$.

Далее, чтобы вычислить $f(n)$, рассмотрим сверху вниз список L для пары (s, n) кончая n . Это соответствует функции Φ_s , для которой $\Phi_s(n) \leq g(n)$. Если такая пара (s, n) существует, полагаем $f(n) = \Phi_s(n)$. С другой стороны, если (s, n) не встречается в L ни при каком s , полагаем $f(n) = 1$. Отметим, что с помощью эффективной процедуры можно определить, встречается или нет в списке L пара (s, n) при определенном s . Это происходит в силу того, что пара (s, n) может встретиться только на шаге $\langle n, m \rangle$, при котором $\Phi_i(n)$ определена и равна m .

По построению если j — любой индекс для f , то $\Phi_j(n) > g(n)$ для почти всех n . Отсюда непосредственно следует часть (1) теоремы. Часть (2) вытекает из доказательства (1), так как это доказательство указывает, как построить f для каждой g : доказательство (1) дает процедуру для перехода от индекса i для g и числа n к значению $f(n)$. При фиксированном i машина, которая выполняет эту процедуру, вычисляет $f(n)$, а индекс этой машины, который легко определить, есть требуемый индекс $\gamma(i)$ для f ; ч. т. д.

Эта последняя теорема используется в качестве леммы в теореме сжимания, которая излагается далее. Фактически для всех машин и кодировок теорема сжимания дает верхнюю и нижнюю границы числа шагов, необходимых для вычисления функции $f(n)$, при этом нижняя граница является частично рекурсивной функцией $\mathcal{E}(n)$, а верхняя граница — общерекурсивной относительно $\mathcal{E}$ функцией $h(n, \mathcal{E}(n))$. В дополнение указывается любопытная связь между мерами вычислений и границами сложности. Эта связь становится более ясной после следующего определения.

Определение. Пусть $\mathcal{M}$ — общерекурсивная функция трех переменных, принимающая значения 0 и 1 и обладающая тем свойством, что для каждого i и n существует не более одного m , такого, что $\mathcal{M}(i, n, m) = 1$. Положим $\mathcal{E}_i(n)$ равным единственному m , если существует m , такое, что $\mathcal{M}(i, n, m) = 1$, и пусть $\mathcal{E}_i(n)$ не определена, если такого m не существует. Множество

$\{\mathcal{G}_i\}$ так определенных частично рекурсивных функций называется измеренным множеством.

Пример 1. Измеренное множество функций есть множество $\{\Phi_i\}$ всех сигнализирующих функций.

Пример 2. Измеренное множество функций $\{\mathcal{G}_i\}$, которое включает функции, вычислимые в реальное время [Ямада, 10], определено так:

$$\mathcal{G}_i(n) = \begin{cases} \varphi_i(n), & \text{если } \varphi_i(n) \text{ определена и } \Phi_i(n) < i\varphi_i(n), \\ \text{не определена} & \text{в противном случае.} \end{cases}$$

Пример 3. Измеренное множество не может включать всех общерекурсивных функций. Для того чтобы показать это, предположим, рассуждая от противного, что $\{\mathcal{G}_i\}$ содержит все общерекурсивные функции. Пусть функция f определена так:

$$f(n) = \begin{cases} 0, & \text{если } \mathcal{M}(n, n, 1) = 1, \text{ т. е. если } \mathcal{G}_n(n) = 1, \\ 1 & \text{в противном случае.} \end{cases}$$

Функция f общерекурсивна, так как общерекурсивна $\mathcal{M}$. Но $f \notin \{\mathcal{G}_i\}$, так как если $f = \mathcal{G}_i$, то $f(i) = 0 \Leftrightarrow \mathcal{G}_i(i) = 1 \Leftrightarrow f(i) = 1$, что противоречиво.

Теорема 8 (сжимания). $\{\mathcal{G}_i\}$ есть измеренное множество $\Leftrightarrow$ существует общерекурсивная функция h , такая, что имеет место следующее: каждой частично рекурсивной функции $\mathcal{G}_i$ соответствует частично рекурсивная функция f , принимающая значения 0 и 1, с той же областью определения, что и $\mathcal{G}_i$, и такая, что (1) если j — любой индекс для f , то $\mathcal{G}_i(n) < \Phi_j(n)$ для почти всех n ; (2) существует такой индекс k для f , что $\Phi_k(n) < h(n, \mathcal{G}_i(n))$ для почти всех n ; (3) существует общерекурсивная функция τ , которая отображает индекс i для любой $\mathcal{G}_i$ в описанный выше индекс k для соответствующей f .

Доказательство. $\Rightarrow$: существует общерекурсивная функция ξ , такая, что $\mathcal{G}_i = \varphi_{\xi(i)}$ для всех i . Доказательство существования функции ξ сходно с проведенным в теореме 2 и поэтому предоставляется читателю.

Пусть γ — функция из теоремы 7, и пусть $f = \varphi_{\gamma\xi(i)}$. Тогда для каждого индекса j функции f теорема 7 гарантирует, что

$$\Phi_j(n) > \varphi_{\xi(i)}(n) (= \mathcal{G}_i(n)),$$

как в (1). Теперь положим

$$p(i, n, m) = \begin{cases} \Phi_{\gamma\xi(i)}(n), & \text{если } \mathcal{G}_i(n) = m, \text{ т. е. если } \mathcal{M}(i, n, m) = 1, \\ 0 & \text{в противном случае.} \end{cases}$$

Определим $h(n+m)=1+\max_{i \leq n} p(i, n, m)$. Тогда h общерекурсивна и $h(n, \mathcal{G}_i(n)) > \Phi_{\gamma\xi(i)}(n)$ для почти всех n , что означает, что так определенная h удовлетворяет (2). Что касается рекурсивности τ в (3), функция $\tau=\gamma\xi$ отображает индекс i для $\mathcal{G}_i$ в нужный индекс $\gamma\xi(i)$ для f . Этим завершается доказательство в прямом направлении.

$\Leftarrow$: Для доказательства того, что $\{\mathcal{G}_i\}$ есть измеренное множество, нужно показать, что $\mathcal{M}$ общерекурсивна. Доказательство того, что $\mathcal{M}(i, n, m)$ общерекурсивна, заключается в процедуре для ее вычисления, которая использует тот факт, что для каждого i и почти всех n

$$h(n, \mathcal{G}_i(n)) > \Phi_{\tau(i)}(n) > \mathcal{G}_i(n).$$

Чтобы вычислить $\mathcal{M}(i, n, m)$, определим сначала верно ли $\Phi_{\tau(i)}(n) < h(n, m)$. Это можно определить, так как h общерекурсивна. Если $\Phi_{\tau(i)}(n) \geq h(n, m)$, то $\mathcal{G}_i(n) \neq m$; поэтому полагаем $\mathcal{M}(i, n, m) = 0$. Если $\Phi_{\tau(i)}(n) < h(n, m)$, то $\Phi_{\tau(i)}(n)$ определена, и так как $\Phi_{\tau(i)}(n) > \mathcal{G}_i(n)$, то определена и $\mathcal{G}_i(n)$. В этом случае возможно определить, имеет ли место $\mathcal{G}_i(n) = m$, и поэтому решить, надо ли полагать $\mathcal{M}(i, n, m)$ равным 0 или 1. Поэтому $\mathcal{M}$ общерекурсивна. Что и требовалось доказать.

В [2] Хартманис и Стирнз доказали красивую специальную форму этой теоремы. Они имели дело с многоленточными машинами, которые печатают произвольное конечное число символов. Будем полагать, что эти машины используют двоичную кодировку входных и выходных данных. Положим также, что $\{\mathcal{G}_i\}$ есть множество функций, вычисляемых в реальное время. Наконец, будем понимать Φ_i как действительное число сдвигов, которое делает i -я машина на своих лентах. При этих условиях теорема гласит, что существует h , приблизительно задаваемая так: $h(n, m) = m^2$. К примеру, пусть $\mathcal{G}_i(n) = 2^{2^n}$. Тогда граница максимального числа шагов, необходимого для вычисления соответствующей f ($f = \varphi_i$), задается так:

$$2^{2^n} < \Phi_f(n) < 2^{2^n+1}.$$

Это означает, что $f(n)$ может быть вычислена на многоленточной машине менее чем за 2^{2^n+1} шагов, но не может быть вычислена менее чем за 2^{2^n} шагов вне зависимости от того, как много дополнительных символов, лент или состояний добавлено к машине.

Наше доказательство теоремы сжимания конструктивно. Поэтому оно может быть использовано для того, чтобы найти h для любого класса машин и любого измеренного множества $\{\mathcal{G}_i\}$. Первое следствие — случай применения этой теоремы. Его

доказательство прямо следует из доказательства теоремы 8. Таким образом иллюстрируется польза теоретических границ, данных здесь.

Следствие 1. Пусть $\{Z_i\}$ — класс одноленточных машин, которые могут печатать и стирать, алфавит которых состоит из символов b (пусто), 0 и 1 и которые используют двоичную кодировку входных и выходных данных. Пусть $\mathcal{G}_i(n) = \Phi_i(n)$ — число шагов, которое делает Z_i на входе n . Тогда функция h из теоремы 8 общерекурсивна и ее значения определяются так: $h(n, m) < (n + m)^7$. Это означает, что каждой сигнализирующей функции Φ_i соответствует частично рекурсивная функция f , принимающая значения 0 и 1, с той же областью определения, что и Φ_i , и такая, что (1) если j есть любой индекс для f , то для почти всех n $\Phi_i(n) < \Phi_j(n)$, (2) функция f имеет индекс $\tau(i)$, такой, что для почти всех n $\Phi_i(n) < \Phi_{\tau(i)}(n) < [n + \Phi_i(n)]^7$.

Доказательство. Так как $\mathcal{M} = M$ — мера вычисления, то она общерекурсивна. Поэтому, согласно теореме сжатия, существует функция h и она общерекурсивна. Доказательство теоремы 8 дает набросок для определения функции h , указанной в этом следствии. При восполнении деталей этого доказательства получается ограничение на h . Восполнение деталей этого доказательства не трудное, но утомительное. Прежде всего функция h определяется в доказательстве теоремы 8 через функцию

$$p(i, n, m) = \begin{cases} \Phi_{\tau(i)}(n), & \text{если } \Phi_i(n) = m, \\ 0 & \text{в противном случае,} \end{cases}$$

где τ представляет функцию $\gamma\xi$ из этого доказательства. Далее

$$\begin{aligned} h(n, m) &= 1 + \max_{i \leq n} p(i, n, m) \leq \\ &\leq 1 + \max_i \{\Phi_{\tau(i)}(n) \mid i \leq n \text{ и } \Phi_i(n) = m\}. \end{aligned}$$

Итак, проблема нахождения верхней границы для $h(n, m)$ сводится к нахождению верхней границы для $\Phi_{\tau(i)}(n)$. Для того чтобы упростить вычисления, сначала находится верхняя граница для $\hat{\Phi}_{\tau(i)}(n)$, которая равна числу шагов, необходимых для вычисления $f(n)$ на 10-ленточной машине. Затем значение $\Phi_{\tau(i)}(n)$ определяется по значению $\hat{\Phi}_{\tau(i)}(n)$. Хартманис и Стирнз [2] показали, что $\Phi_{\tau(i)}(n) \leq [\hat{\Phi}_{\tau(i)}(n)]^2$ в случае одноленточных и 10-ленточных машин, печатающих и стирающих произвольно большое количество символов. Машины, рассматриваемые в этом следствии, могут, однако, печатать и стирать только пустой символ, 0 и 1, и по этой причине на самом деле выполняется

отношение $\Phi_{\tau(i)}(n) \leq 10[\hat{\Phi}_{\tau(i)}(n)]^2$, где 10 соответствует 10 лентам 10-ленточной машины.

Вычисление $f(n)$ описывается в доказательстве теоремы 7. Далее следует повторение этап за этапом части доказательства, касающейся конструкции, вместе с утверждениями, заключенными в скобки, которые дают число шагов, необходимое для проведения каждого из этапов вычисления. Символ c всюду используется для обозначения константы. Чтобы вычислить $f(n)$:

А. Вычислим $\Phi_i(n) = m$ и перейдем к шагу $0 = \langle 0, 0 \rangle$. Число шагов, необходимое, чтобы сделать это, составляет $[\alpha \leq c + c \log n + cm \log m]$. Далее индуктивно

В. На шаге $x = \langle p, q \rangle$

1. Определим, имеет ли место $\Phi_i(p) = q$ $[\beta_1 \leq c + c \log p + cq \log q]$.

Предположим, что $\Phi_i(p) = q$.

2. Выделим индексы $s_1, \dots, s_t$ всех невычеркнутых функций, принадлежащих множеству $\{\varphi_0(p), \dots, \varphi_p(p)\}$ $[\beta_2 \leq c(p+1)(p+q)(\log(p+q))]$.

3. Выберем наименьший индекс s_k , такой, что $\Phi_{s_k}(p) < q$, и присоединим (s_k, p) к списку. Нам требуется определить следующие величины:

$s(i) =_{df}$ — число шагов, необходимое для того, чтобы напечатать программу машины Z_i ;

$$r(p) =_{df} \sum_{i=0}^p s(i) \quad [r(p) < cp^2 \log p];$$

$$d(p) =_{df} \sum_{i=0}^p d_i, \text{ где } d_i \Phi_i(n) \text{ есть число шагов, необходимое для того, чтобы смоделировать вычисление } Z_i \text{ на входе } n \quad [d(p) < cp(\log p)^2];$$

$$[\beta_3 \leq r(p) + cd(p)(q \log q) + cpq].$$

С. Перейдем от шага $n = \langle p, q \rangle$ к шагу $n+1$, если $\langle p, q \rangle \neq \langle n, m \rangle$ $[\gamma \leq c \log p + c \log q + c]$.

Д. Если ни одна из функций $\Phi_s(n)$ не вычеркнута на шаге $\langle n, m \rangle$, полагаем $f(n) = 0$. В противном случае $\Phi_s(n)$ вычеркивается на шаге $\langle n, m \rangle$ и поэтому полагаем $f(n) = 1 - \varphi_s(n)$ $[\delta \leq c]$.

Общее число шагов, требуемых для вычисления $f(n)$ на 10-ленточной машине, составляет

$$\hat{\Phi}_{\tau(i)}(n) \leq \left\{ \alpha + \sum_{q=0}^{n+m} \sum_{p=0}^{n+m-q} (\gamma + \beta_1) + (\beta_2 + \beta_3) |_{p+q=n+m} \cdot (1+n+m) + \delta \right\} <$$

$$< c + c[\log(1+n+m)]^3 \cdot (1+n+m)^3.$$

Но $\Phi_{\tau(i)}(n) \leq 10 [\hat{\Phi}_{\tau(i)}(n)]^2$. Поэтому допустимая граница для h такова: $h(n, m) < (n+m)^7$. Что и требовалось доказать.

Определение. Утверждение «существуют сколь угодно большие общерекурсивные функции r со свойством A » означает, что каждой общерекурсивной функции s соответствует r со свойством A такая, что $r(n) > s(n)$ для всех n . К примеру, существуют сколь угодно большие сигнализирующие функции (теорема 4).

Определение. Сказать, что все достаточно большие общерекурсивные функции r обладают свойством A , значит сказать, что существует общерекурсивная функция s , такая, что если $r(n) > s(n)$ для всех n , то r обладает свойством A . К примеру, все достаточно большие общерекурсивные функции r делают по крайней мере 2^n шагов при вычислении для почти всех n (это следует из теоремы 3 и того факта, что обычная машина Тьюринга, сигнализирующая функция которой действительно считает число шагов, делает 2^n шагов только для того, чтобы напечатать значение $r(n)$).

Следствие 2. Существуют сколь угодно большие общерекурсивные функции r и связанная с каждой r общерекурсивная функция f , принимающая значения 0 и 1, такие, что (1) если i есть любой индекс для f , то $\Phi_i(n) > r(n)$ для почти всех n ; (2) существует индекс k для f , такой, что $r(n+1) > \Phi_k(n) > r(n)$ для почти всех n .

Доказательство этого следствия мы предоставляем читателю.

Можно предположить, что всем достаточно большим общерекурсивным функциям h соответствуют общерекурсивные функции f , такие, что (1) если i есть любой индекс для f , то $h(n) < \Phi_i(n)$ для почти всех n ; (2) существует индекс k для f , такой, что $h(n) < \Phi_k(n) < h(h(n))$ для почти всех n . Но это ложно, если нет некоторого условия, связывающего h и измеренное множество. Действительно, существует произвольно большая общерекурсивная функция h , такая, что для каждого индекса j или $\Phi_j(n) < h(n)$, или $h[h(n)] < \Phi_j(n)$ для бесконечно многих n . К сожалению, функция может обладать этим свойством и в то же время может иметь место слабая форма этого сжимания. Можно показать, что каждой достаточно большой общерекурсивной функции h соответствует общерекурсивная функция f , такая, что (1) если i есть любой индекс для f , то для бесконечно многих n $h(n) < \Phi_i(n)$; (2) для некоторого индекса k функции f имеем $h(n) < \Phi_k(n) < h[h(n)]$ для бесконечно многих n .

Чтобы доказать это утверждение, докажем, во-первых, что каждой достаточно большой общерекурсивной функции h соответствует индекс j , такой, что $h(n) < \Phi_j(n) < h[h(n)]$ для бесконечно многих n . Далее, используя теорему сжатия, получаем нужный результат.

ЛИТЕРАТУРА

1. Arbib M. A., Blum M., Machine dependence of degrees of difficulty, *Proc. Amer. Math. Soc.*, **16**, № 3 (June 1965), 442—447.
2. Hartmanis J., Stearns R. E., On the computational complexity of algorithms, *Trans. Amer. Math. Soc.*, **117**, № 5 (May 1965), 285—306. (Русский перевод: Кибернетический сборник, новая серия, вып. 4, 1967.)
3. Hartmanis J., Stearns R. E., Computational complexity of recursive sequences, *Proc. 5th Annual Symp. on Switching Theory and Logical Design*, Princeton, N. J., 1964.
4. Myhill J., Linear bounded automata, WADD Tech. Note 60-165, U. of Pennsylvania, Rep. № 60-22 (June 1960).
5. Rabin M. O., Degree of difficulty of computing a function and a partial ordering of recursive sets, Tech. Rep. № 2, Hebrew U., Jerusalem, Israel, April 1960.
6. Rabin M. O., Real time computation, *Israel J. Math.*, **1** (1963), 203—211. (Русский перевод см. в наст. сб., стр. 156—167.)
7. Ritchie R. W., Classes of predictably computable functions, *Trans. Amer. Math. Soc.*, **106**, № 1 (1963), 139—173. (Русский перевод см. в наст. сб., стр. 50—93.)
8. Rogers H., Jr., Gödel numberings of partial recursive functions, *J. Symbolic Logic*, **23**, № 3 (1958), 331—341.
9. Rogers H., Jr., Recursive functions and effective computability, McGraw-Hill, New York, 1967; готовится к печати русский перевод.
10. Yamada H., Real-time computation and recursive functions not real-time computable, *IRE Trans.*, EC-11 (1962), 753—760. (Русский перевод см. в наст. сб., стр. 139—155.)
11. Cobham A., The intrinsic computational complexity of functions, *Proc. 1964 Int. Congress on Logic, Methodology and Philosophy of Science*, North-Holland, Amsterdam, 1965, 24—30.
12. Cooke S. A., On the minimum computation time of functions, Bell Labs Rep. BL-41, 1966.
13. Winograd S., On the time required to perform multiplication, IBM Res. Rep. RC-1504, 1966 (Русский перевод: Кибернетический сборник (новая серия), вып. 6, 1969.)

ОБ ОБЪЕМЕ МАШИН¹⁾

М. Блюм

ВВЕДЕНИЕ

Под машинами мы подразумеваем машины Тьюринга, идеализированные программы вычислительных машин или любые другие идеализированные устройства для вычисления рекурсивных функций. Машины и их объем должны удовлетворять аксиомам разд. 1. В разд. 3 введен общий ряд аксиом для вычисления по шагам. Они определяют, какие определения «шага» вычисления допускаются. Все эти аксиомы так необычайно слабы, что им удовлетворяют любая разумная модель вычислительной машины и любое разумное определение объема и шага.

Все наши примеры относятся к специальному классу машин и специфическому понятию объема и шага: это класс одноленточных двухсимвольных машин Тьюринга, работающих в унарной системе счисления и определяемых множеством четверок (команд); Дэвис [5]. Объем машины Тьюринга полагается равным числу команд, определяющих машину. За один шаг вычисления принимается сдвиг ленты на один квадрат (влево или вправо), или один акт печати, или одно стирание.

В отличие от примеров наши теоремы относятся к любым устройствам и любым мерам объема и шага вычисления, которые удовлетворяют аксиомам.

1. Натуральные числа суть $N=0, 1, 2, \dots$. Пусть (φ_i) с i , пробегаящим N , обозначает рекурсивно перечислимую (р. п.) последовательность всех частично рекурсивных (ч. р.) функций одного переменного. Пусть (φ_i^k) , где k — фиксированное целое число, $k \geq 2$, обозначает рекурсивно перечислимую последовательность всех ч. р. функций k переменных.

Определение (см. Роджерс [8]). Последовательность (φ_i) имеет допустимую гёделевскую нумерацию тогда и только тогда, когда она удовлетворяет теореме об универсальной машине Тьюринга и теореме об итерации.

¹⁾ Blum M., On the size of machines, *Information and Control*, 11, № 3 (1967), 257—265.

Теорема (об универсальной машине Тьюринга). *Существует частично рекурсивная функция f , такая, что $f(i, x) = \varphi_i(x)$ для всех i и x .*

Теорема (об итерации). *Существует рекурсивная функция s , такая, что $\varphi_i^2(x, y) = \varphi_{s(i, x)}(y)$ для всех i, x, y .*

Теорема (Роджерса, см. [8]). *Если две последовательности (φ_i) и (ψ_i) имеют допустимые гёделевские нумерации, то каждая из этих последовательностей является рекурсивной перестановкой другой, т. е. существует взаимно однозначная рекурсивная функция $f: N \rightarrow N$, такая, что $\varphi_{f(i)} = \psi_i$ для всех i .*

Эта важная теорема позволяет получить, как мы увидим далее, следствия, касающиеся объема машины.

Предположим, что каждая функция φ_i вычислима посредством некоторого множества команд M_i , M_i может быть машиной Тьюринга, программой идеализированной вычислительной машины или системой уравнений. Скажем, что последовательность (M_i) имеет допустимую гёделевскую нумерацию, если таковую имеет (φ_i) ; далее мы везде будем предполагать, что (φ_i) такова. Теорема Роджерса позволяет нам предполагать (так мы и сделаем без потери общности), что любые два класса машин (M_i) и (T_i) (оба с допустимыми гёделевскими нумерациями) упорядочены так, что M_i и T_i вычисляют одну и ту же функцию φ_i .

Аксиомы. *Рекурсивная функция $\|$, отображающая натуральный ряд N (рассматриваемый как множество индексов) в N (рассматриваемый как множество объемов), называется мерой объема машин (при этом $|i|$ называется объемом машины M_i) тогда и только тогда, когда существуют*

(1) *самое большее конечное число машин любого данного объема и*

(2) *эффективная процедура, позволяющая для всякого y определять, какие машины имеют объем y .*

Предположим, что мы сравниваем машины из одного класса (M_i) с машинами из другого класса (T_i) . Например, (M_i) может быть классом одноленточных двухсимвольных машин Тьюринга, тогда как (T_i) может быть классом 10-ленточных 10-символьных машин Тьюринга. Следующее предложение показывает, что объем M_i приблизительно равен объему T_i . Это немедленно следует из теоремы Роджерса:

Предложение 1. *Пусть $\|_M$ и $\|_T$ — две меры объема машин (M_i) и (T_i) соответственно. Тогда существует такая рекурсивная функция g , что для всех i*

- 1) $|i|_M \leq g|i|_T$ (т. е. не может быть много больше чем T_i),
- 2) $g|i|_M \geq |i|_T$ (т. е. не может быть много больше чем M_i).

Доказательство. Пусть $g(x) = \max\{g_1(x), g_2(x)\}$, где g_1 определена так, что $|i|_M \leq g_1|i|_T$, и g_2 определена так, что $g_2|i|_M \geq |i|_T$. Для вычисления $g_1(x)$ заметим, что множество $\{i \mid |i|_T = x\}$ конечно и может быть эффективно указано. Положим $g_1(x) = \max\{|i|_M \mid i \text{ удовлетворяет } |i|_T = x\}$. Пусть $g_2(x) = \max\{|i|_T \mid i \text{ удовлетворяет } |i|_M = x\}$. Доказательство завершено.

Если M_i много меньше чем M_j , то тогда можно подозревать, что T_i меньше чем T_j . Это является содержанием следующего предложения.

Предложение 2. Существует такая рекурсивная функция h , что для всех i, j :

$$1) |i|_M \leq |j|_M \Rightarrow |i|_T \leq h|j|_T, \quad 2) h|i|_M \leq |j|_M \Rightarrow |i|_T \leq |j|_T.$$

Доказательство. Заметим сразу, что функцию g в предложении 1 можно сделать монотонно возрастающей, положив

$$g(x) = 1 + \max\{g_1(x), g_2(x), g(x-1)\}.$$

Будем считать, что g монотонно возрастает. Пусть $h(x) = gg(x)$.

1) Если $|i|_M \leq |j|_M$, то $|i|_T \leq g|i|_M$ (согласно п. 2 предложения 1) $\leq g|j|_M$ (в силу монотонного возрастания g) $\leq gg|j|_T$ (ввиду п. 1 предложения 1).

2) Если $h|i|_M \leq |j|_M$, то аналогично устанавливается, что $g|i|_T \leq gg|i|_M = h|i|_M \leq |j|_M \leq g|j|_T$, а поскольку g монотонно возрастает, то $|i|_T \leq |j|_T$.

2. Машина Тьюринга может вычислять постоянную функцию $c_n(x) = n$, запоминая n цифр ответа с помощью внутренних состояний. Вообще говоря, такая машина будет чрезвычайно большой. Например, если $n = 10^{10}$, меньшая машина может дать на выходе 10^{10} при десятикратном умножении числа 10 самого на себя, а не повторением всех 10^{10} цифр. Мы докажем, что любая бесконечная р. п. последовательность машин содержит машины, объем которых может быть уменьшен подобным образом.

Теорема 1. 1) Пусть g — произвольная рекурсивная функция с бесконечным множеством значений (g перечисляет индексы бесконечной последовательности машин).

2) Пусть f — любая рекурсивная функция. Тогда существуют числа $i, j \in N$, равномерные относительно¹⁾ f и g и такие, что

¹⁾ Равномерность числа i относительно функции f означает, что i эффективно вычисляется по f . — Прим. перев.

3) $\varphi_i = \varphi_{g(j)}$,

4) $f|i| < |g(j)|$ (т. е. M_i значительно меньше чем $M_{g(j)}$).

Пример. Если $f(x) = 100x$, то $M_{g(j)}$ в 100 раз больше чем M_i , хотя и M_i и $M_{g(j)}$ вычисляют одну и ту же функцию.

Доказательство. Мы укажем способ определения чисел i и j , равномерных относительно f и g . Рассмотрим следующее множество команд:

«Для входов x и y сначала вычислить $f|y|$.

Затем вычислять $g(0), g(1), \dots$ до тех пор, пока не найдется j , такое, что $f|y| < |g(j)|$.

Тогда вычислить $\varphi_{g(j)}(x)$ и выдать полученное число в качестве выхода».

Эти команды определяют частично рекурсивную функцию $\varphi_z^2(x, y)$, индекс z которой равномерен относительно f и g . По теореме рекурсии¹⁾ найдется число i , равномерное относительно f, g и такое, что

$$\text{уравнение 1: } \varphi_i(x) = \varphi_z^2(x, i)$$

справедливо для всех x .

Покажем, что число i обладает желаемыми свойствами.

Из условий 1, 2 следует, что можно найти число j , равномерное относительно f, g , которое удовлетворяет 4: $f|i| < |g(j)|$.

Сначала находим число i , для которого мы показали, что оно является равномерным относительно f, g . Затем сравниваем $f|i|$ с $|g(0)|, |g(1)|, \dots$ до тех пор, пока не найдем j , удовлетворяющее 4. Для этого j (см. уравнение 1 и ранее указанные команды) выполняется условие 3: $\varphi_i(x) = \varphi_z^2(x, i) = \varphi_{g(j)}(x)$ для всех x .

Функция g является алгоритмической функцией, перечисляющей бесконечное множество машин. Конструктивность этого доказательства позволяет нам эффективно заменить функцию g на функцию g' , перечисляющую машины, которые по объему не больше, а иногда значительно меньше, чем соответствующие машины, перечисляемые g .

Могут сказать, что, поскольку практически все вычислимые функции примитивно рекурсивны, нет необходимости в общей рекурсии для практических целей. Однако теорема 1 дает практический аргумент в пользу общей рекурсии: из нее следует, что существуют примитивно рекурсивные функции, наименьшая сложность которых (определяемая уравнениями) в примитивно

¹⁾ Теорема рекурсии гласит, что для всякой частично рекурсивной функции h существует целое число i , равномерное относительно h и такое, что $\varphi_i(x) = h(x, i)$ для всех x .

рекурсивной форме значительно больше, чем их наименьшая сложность в общерекурсивной форме. Точнее говоря, предположим, что примитивная и общая рекурсии определены схемами, как у Дэвиса [5]. Объемом (сложностью) схемы объявим число входящих в нее букв. Тогда у каждой примитивно рекурсивной функции будет по крайней мере одно наименьшее представление в виде примитивно рекурсивной схемы. Множество всех таких наименьших представлений будет р. п., пусть их пересчитывает функция g . Полагая введенную выше функцию $f(x) = n \cdot x$, по теореме 1 найдем примитивно рекурсивную функцию, наименьшая примитивно рекурсивная схема которой в n или более раз больше, чем ее наименьшая общерекурсивная схема. Методом теоремы 2 можно показать, что примитивно рекурсивная схема и намного более простая общерекурсивная схема требуют приблизительно одинакового числа шагов при вычислении одной и той же функции.

3. Если машина минимизирована в смысле объема, то часто случается, что меньшая машина требует больше шагов, чем исходная, бóльшая по объему машина. Подготавливая путь для изучения этого обстоятельства, мы теперь введем аксиомы Блума [10] для вычисления. Каждой частично рекурсивной функции φ_i сопоставляется другая частично рекурсивная функция Φ_i , называемая функцией, оценивающей сложность вычисления, или сигнализирующей функцией. Интуитивно $\Phi_i(x)$ представляет число шагов, которые делает машина M_i с входом x , вычисляя $\varphi_i(x)$.

Аксиомы. Предположим, что последовательность всех ч. р. функций (φ_i) имеет допустимую гёделевскую нумерацию. Пусть (Φ_i) обозначает любую р. п. последовательность (некоторых, но не обязательно всех) ч. р. функций. Скажем, что (Φ_i) является последовательностью сигнализирующих функций для (φ_i) тогда и только тогда, когда

1) $\varphi_i(x)$ определена тогда и только тогда, когда определена $\Phi_i(x)$;

2) существует рекурсивная функция R , такая, что

$$R(i, x, y) = \begin{cases} 1, & \text{если } \Phi_i(x) = y, \\ 0, & \text{если } \Phi_i(x) \neq y \end{cases}$$

(т. е. график функции $\Phi_i(x)$ должен быть рекурсивным).

Первая аксиома утверждает, что если $\varphi_i(x)$ определена, то число шагов, необходимых для ее вычисления, конечно, и, обратно, если число шагов, которые надо сделать, чтобы вычислить $\varphi_i(x)$, конечно, то функция $\varphi_i(x)$ определена. Мы будем записы-

вать « $\Phi_i(x) < \infty$ » вместо « $\Phi_i(x)$ определена» и « $\Phi_i(x) = \infty$ » вместо « $\Phi_i(x)$ неопределена». Вторая аксиома утверждает, что существует алгоритм для решения вопроса о том, заканчивается ли M_i вычисление с входом x за y шагов. Предикат R этой аксиомы аналогичен предикату T Клини.

Заметим, что аксиомы 1 и 2 независимы: последовательность сигнализирующих функций, удовлетворяющих аксиоме 1, но не удовлетворяющих аксиоме 2, получится, если положить $\Phi_i = \varphi_i$ для всех i . С другой стороны, $\Phi_i(x) = 0$ для всех i , x удовлетворяет аксиоме 2, но не удовлетворяет аксиоме 1.

В следующих разделах мы везде будем предполагать, что (Φ_i) есть последовательность сигнализирующих функций для (φ_i) . Функция R не будет нигде встречаться, хотя она неявно используется во всяком предложении вида «проверим, верно ли, что $\Phi_i(x) = y$ ». Эта проверка осуществляется посредством вычисления $R(i, x, y)$.

4. Следующая теорема, обобщающая теорему 1, показывает, что увеличение числа шагов, которое может иметь место при приведении машины, т. е. при переходе к машине меньшего объема, незначительно для тех функций, которые вычисляются достаточно сложно.

Теорема 2. *Существует рекурсивная функция h , такая, что если*

1) g — любая рекурсивная функция с бесконечным множеством значений;

2) f — любая рекурсивная функция,

то существуют $i, j \in \mathbb{N}$, равномерные относительно f и g и такие, что

3) $\varphi_i = \varphi_{g(j)}$;

4) $f|i| < |g(j)|$;

5) для почти всех x , т. е. для всех x , кроме, быть может, конечного числа, $[\varphi_{g(j)}(x) \text{ определена} \Rightarrow \Phi_i(x) \leq h(x, \varphi_{g(j)}(x))]$ (т. е. M_i не делает намного больше шагов, чем $M_{g(j)}$).

Замечание. Функция $h(x, y) = c(x + y)$, где $c = \text{const}$, подходит для класса машин Тьюринга, определенных во введении. Если эти машины Тьюринга имеют неограниченное число символов на ленте, то можно выбрать $c = 1$.

Доказательство. Определим рекурсивную функцию r . Эта функция r обладает тем свойством, что для любых f, g , удовлетворяющих 1, 2, существуют целые числа i, j , которые определены фактически в доказательстве теоремы 1, и целое число ω , такие, что

(а) для всякого x если $\varphi_{g(j)}(x)$ определена, то $r(\omega, x, \Phi_{g(j)}(x)) = \Phi_i(x)$.

С помощью r определим рекурсивную функцию h :

$$h(x, y) = \max \{r(\omega, x, y) \mid \omega \leq x\},$$

откуда следует, что для достаточно больших x

(б) если $\varphi_{g(j)}(x)$ определена, то $h(x, \Phi_{g(j)}(x)) \geq r(\omega, x, \Phi_{g(j)}(x))$.

Из (а) и (б) следует п. 5 теоремы 2.

Определение $r(\omega, x, y)$. Выберем некоторое фиксированное взаимно однозначное отображение $N \rightarrow \underbrace{N \times \dots \times N}_{5 \text{ раз}}$.

Число ω есть код пятерки слов $\langle \omega_1, \dots, \omega_5 \rangle$.

Пусть $f = \varphi_{\omega_1}$, $g = \varphi_{\omega_2}$. Определим i с помощью f и g , используя алгоритм из доказательства теоремы 1. Это возможно, потому что этот алгоритм определяет i для любых ч. р. f, g , не обязательно удовлетворяющих 1, 2 (хотя j может быть неопределено, если п. 1 и 2 не удовлетворяются).

Пусть теперь $r(\omega, x, y) = \Phi_i(x)$, если выполняются следующие требования:

(1) $\Phi_{\omega_1} |i| = \omega_3$ (так определяется $f |i|$),

(2) $\sum_{k=0}^{\omega_4} \Phi_{\omega_2}(k) = \omega_5$ (так определяются $g(0), \dots, g(\omega_4)$),

(3) существует число j , такое, что $j \leq \omega_4$ и $f |i| < |g(j)|$,

(4) для наименьшего j , удовлетворяющего (3), имеет место $\Phi_{g(j)}(x) = y$.

В противном случае положим $r(\omega, x, y) = 0$.

Если (1) — (4) выполняются, то Φ_i и $\Phi_{g(j)}$ имеют одну и ту же область определения и $\Phi_{g(j)}(x)$ определяется согласно (4). Заметим, что если f, g удовлетворяют 1, 2, то существует такое j , что $|g(j)| > f |i|$. Поэтому существует $\omega = \langle \omega_1, \dots, \omega_5 \rangle$, где ω_1, ω_2 суть индексы f и g соответственно, которые удовлетворяют (1) — (3). Для этого ω выполняется равенство (а). Ч. т. д.

Можно показать, что теорема 2 становится неверной, если заменить в п. 5 выражение «для почти всех x » на «для всех x ».

В то же время теорема 3, весьма сходная с теоремой 2, содержит в п. 5 выражение «для всех x ».

Теорема 3. 1) Пусть g — рекурсивная функция, удовлетворяющая условию $\varphi_{g(j)} = \varphi_j$ для всех j .

2) Пусть f и h — рекурсивные функции.

Тогда для всякого $i \in N$ существует $j \in N$, равномерное относительно f, h, g, i и такое, что

3) $\varphi_i = \varphi_{g(j)}$.

4) $f |i| < |g(j)|$ (т. е. M_i значительно меньше чем $M_{g(j)}$).

5) Для всех x из того, что $\varphi_i(x)$ определена, следует, что $h(x, \Phi_i(x)) < \Phi_{g(j)}(x)$ (т. е. M_i делает значительно меньше шагов чем $M_{g(j)}$).

Доказательство. Выберем любое число i . Мы ищем такое j , для которого имеют место п. 3—5. Это j определяется следующим образом: определим частично рекурсивную функцию $r(n, x, y)$, равномерную относительно f, h, g, i (равенство 1). По теореме рекурсии получаем рекурсивную функцию q , такую, что $\varphi_{q(n)}(x) = r(n, x, q(n))$ (равенство 2). Эта функция q равномерна относительно f, h, g, i . Наконец, мы покажем, как выбрать эффективно n так, чтобы $j = q(n)$ удовлетворяло п. 3—5.

Равенство 1. $r(n, x, y) = n$, если $f|i| \geq |g(y)|$ (т. е. если п. 4 не выполнен при $y = j$), и $r(n, x, y) = \varphi_i(x)$, если

(1) $f(i) < |g(y)|$ (п. 4 имеет место для $y = j$),

(2) $\varphi_i(x)$ определена,

(3) $h(x, \Phi_i(x)) < \Phi_{g(y)}(x)$ (п. 5 выполняется при $y = j$).

В остальных случаях $r(n, x, y)$ не определена.

Функция r вычислима. Теорема рекурсии¹⁾ дает нам рекурсивную функцию q , равномерную относительно f, h, g, i , такую, что

$$\varphi_{q(n)}(x) = r(n, x, q(n)).$$

Исключая r , мы получаем

Равенство 2. $\varphi_{q(n)}(x) = n$, если $f|i| \geq |gq(n)|$,

и $\varphi_{q(n)}(x) = \varphi_i(x)$, если

(1) $f(i) < |gq(n)|$,

(2) $\varphi_i(x)$ определена,

(3) $h(x, \Phi_i(x)) < \Phi_{gq(n)}(x)$.

В остальных случаях $\varphi_{q(n)}(x)$ не определена.

Как выбрать n ? Заметим сначала, что $f|i| \geq |gq(n)|$ означает, что существует машина, которая вычисляет константную функцию $\varphi_{gq(n)}(x) = \varphi_{q(n)}(x) = n$ и объем которой меньше некоторого фиксированного целого числа $f|i|$. Это невозможно для всех n , поэтому существует такое n , что $|f(i)| < |gq(n)|$. Чтобы найти это значение n , мы просто проверяем $q(0), q(1), q(2), \dots$ до тех пор, пока не найдем n , для которого $|f(i)| < |gq(n)|$. При таком способе выбора n положим $j = q(n)$. Таким образом п. 4 удовлетворяется. Отсюда следует

¹⁾ Расширенная теорема рекурсии утверждает, что для каждой частично рекурсивной функции $r(x_1, \dots, x_m, y_1, \dots, y_n, z)$ существует рекурсивная функция q , равномерная относительно r и такая, что

$$\varphi_q(x_1, \dots, y_m)(y_1, \dots, y_n) = r(x_1, \dots, x_m, y_1, \dots, y_n, q(x_1, \dots, x_m)).$$

Равенство 3. $\varphi_j(x) = \varphi_i(x)$, если

(2) $\varphi_i(x)$ определена,

(3) $h(x, \Phi_i(x)) < \Phi_{g(j)}(x)$.

В противном случае $\varphi_j(x)$ неопределена.

Выполняется п. 5. Предположим, рассуждая от противного, что $\varphi_i(x)$ определена, но при этом

$$h(x, \Phi_i(x)) \geq \Phi_{g(j)}(x).$$

Тогда $r(n, x, j)$ неопределена. Отсюда $\varphi_j(x) \neq \varphi_{g(j)}(x)$, так как $\Phi_{g(j)}(x) \leq h(x, \Phi_i(x)) < \infty$. Это противоречит 2: $\varphi_{g(j)} = \varphi_j$.

Выполняется п. 3. Если $\varphi_i(x)$ неопределена, то неопределена и $\varphi_j(x)$ (равенство 3). Таким образом, $\varphi_{g(j)}(x) = \varphi_j(x) = \varphi_i(x)$. Если $\varphi_i(x)$ определена, то из доказательства п. 5 вытекает, что $h(x, \Phi_i(x)) < \Phi_{g(j)}(x)$, т. е. (2), (3) в равенстве 3 удовлетворяются, а поэтому $\varphi_{g(j)}(x) = \varphi_i(x)$, ч. т. д.

Любую алгоритмическую функцию g для уменьшения объема машин можно эффективно заменить на другую функцию g' , которая дальше уменьшает объем машин и длину вычисления для бесконечного числа машин. Любопытно, что теорема 3 симметрична относительно объема машины (п. 4) и длины вычислений (п. 5).

ЛИТЕРАТУРА

1. Arbib M., Speed-up theorems and incompleteness theorems in automata theory, «Automata theory» (E. Caianiello, ed.) New York, 1966, 6—24.
2. Arbib M., Blum M., Machine dependence of degrees of difficulty, *Proc. AMS*, 16, № 3 (1965), 442—447.
3. Chaitin G., On the length of programs for computing finite binary sequences, *J. ACM*, 13, № 4 (1966), 547—569.
4. Chaitin G., On the simplicity and speed of programs for computing infinite sets of natural numbers, *J. ACM*, 16, № 3 (1969) 407—422.
5. Davis M., Computability and unsolvability, McGraw-Hill, New York, 1958.
6. Hartmanis J., Stearns R., On the computational complexity of algorithms, *Trans. AMS*, 117, № 5 (1965), 285—306. (Русский перевод: Кибернетический сборник (новая серия), вып. 4, «Мир», 1967, стр. 57—85.)
7. Minsky M., Size and structure of universal Turing machines using tag systems, «Recursive Function Theory, Proc. of the Symp. in Pure Mathematics» (Amer. Math. Soc. Publ.), vol. V 229—237, 1962.
8. Rogers H., Jr., Gödel numberings of partial recursive functions, *J. Symb. Log.*, 23, № 3 (1958), 331—341.
9. Rogers H., Jr., Recursive functions and effective computability, McGraw-Hill, New York, 1967 готовится к печати русский перевод
10. Blum M., A machine-independent theory of the complexity of recursive functions, *J. ACM*, 14, № 2 (1967), 322—336. (Настоящий сборник, стр. 401—422.)

СОДЕРЖАНИЕ

Предисловие	5

I. КЛАССИФИКАЦИЯ РЕКУРСИВНЫХ ФУНКЦИЙ

А. Гжегорчик. Некоторые классы рекурсивных функций. <i>Перевод А. А. Мучника</i>	9
Р. В. Ричи. Классы предсказуемо вычислимых функций. <i>Перевод Н. Н. Катериночкиной</i>	50
Дж. П. Клив. Иерархия примитивно рекурсивных функций. <i>Перевод С. С. Марченкова</i>	94
П. Акст. Итерация примитивной рекурсии. <i>Перевод А. Гагарина</i> . . .	114
П. Акст. Итерация относительной примитивной рекурсии. <i>Перевод С. Каллибекова</i>	118
Добавление редактора. О двух подходах к классификации рекурсивных функций	123

II. О СЛОЖНОСТИ ВЫЧИСЛЕНИЙ НА МАШИНАХ ТЬЮРИНГА

Х. Ямада. Вычисления в реальное время и рекурсивные функции, не вычислимые в реальное время. <i>Перевод В. Е. Фельдмана</i>	139
М. Рабин. Вычисления в реальное время. <i>Перевод В. Е. Фельдмана</i> .	156
А. Розенберг. Языки, определяемые в реальное время. <i>Перевод М. Афанасьева и М. В. Ломковской</i>	168
Ф. К. Хенни, Р. Е. Стирнз. Моделирование многоленточной машины Тьюринга на двуленточной. <i>Перевод Ю. А. Бухштаба</i>	194
С. С. Раби, П. К. Фишер. Методы перевода и сложность вычислений. <i>Перевод А. А. Мучника</i>	213
Ф. К. Хенни. Вычисления на одноленточной машине Тьюринга с записью на ленте. <i>Перевод Ю. Я. Брейтбарта</i>	223
Ф. К. Хенни. Вычисления на машинах Тьюринга со входом. <i>Перевод А. Набибина</i>	249
П. Стрнад. Распознавание на машине Тьюринга со входом. <i>Перевод Ю. А. Бухштаба</i>	271
Дж. Хартманис. Сложность вычислений на одноленточных машинах Тьюринга. <i>Перевод А. А. Мучника</i>	282
Р. Е. Стирнз, Дж. Хартманис, П. М. Льюис II. Иерархии вычислений с ограниченной памятью. <i>Перевод М. И. Кановича</i>	301
П. М. Льюис II, Р. Е. Стирнз, Дж. Хартманис. Границы памяти для разрешения контекстно-свободных и контекстных языков. <i>Перевод М. И. Кановича</i>	320
Дж. Хартманис. Об объеме памяти, необходимом для распознавания бесконтекстных языков. <i>Перевод М. В. Ломковской</i>	339
Д. Х. Янгер. Распознавание и анализ контекстно-свободных языков за время n^3 . <i>Перевод В. Е. Фельдмана</i>	344
Т. Касами. О времени машинного распознавания языков, порожденных линейными грамматиками. <i>Перевод В. Е. Фельдмана</i>	363
Добавление к статье Касами. Распознавание металинейных языков по методу Т. Касами	369
М. Фишер, А. Розенберг. Решение задачи о пересечении начала координат в реальное время. <i>Перевод В. Е. Фельдмана</i>	372
П. Фишер, А. Мейер, А. Розенберг. Счетчиковые машины и счетчиковые языки. <i>Перевод В. Е. Фельдмана</i>	380
М. Блюм. Машинно-независимая теория сложности рекурсивных функций. <i>Перевод В. А. Козмидиади</i>	401
М. Блюм. Об объеме машин. <i>Перевод А. А. Мучника</i>	423