Кибернетический сборник

НОВАЯ СЕРИЯ

ВЫПУСК

12

Сборник переводов

Под редакцией

О. Б. ЛУПАНОВА

ИЗДАТЕЛЬСТВО „МИР“

Москва 1975

К 519.95

Научный совет по кибернетике
Академии наук СССР

Двенадцатый выпуск новой серии кибернетических сборников содержит переводы статей по современным проблемам теоретической кибернетики. Статьи подразделены на четыре части: «Математические вопросы», «Вычислительные машины и программирование», «Вопросы информационного поиска», «Математическая биология». Особый интерес представляют статьи С. Кука, Л. Мейера, Дж. Хапкрофта по математическим вопросам, статья Сэлтона по информационному поиску и статья У. Хоффмана по аксиоматике математической биологии.

Сборник рассчитан на научных работников, инженеров, аспирантов и студентов различных специальностей, занимающихся и интересующихся теоретической кибернетикой и ее приложениями.

Редакция литературы по математическим наукам

20205—021
041(01)—75 21—75

Сложность процедур вывода теорем[1])

С. А. Кук

Показано, что любая проблема распознавания, решаемая на недетерминированной машине Тьюринга за полиномиально ограниченное время, может быть «сведена» к проблеме распознавания того, является ли некоторая пропозициональная формула тавтологией. Здесь «сведена» означает, грубо говоря, что первая проблема может быть решена детерминированно за полиномиальное время при условии, что имеется оракул для решения второй. Исходя из этого понятия сводимости, определены полиномиальные степени трудности и показано, что проблема определения тавтологичности имеет ту же самую полиномиальную степень, что и проблема определения того, верно ли, что первый из двух заданных графов изоморфен некоторому подграфу второго. Обсуждаются и другие примеры. Вводится и обсуждается также метод измерения сложности процедур вывода для исчисления предикатов.

Везде в этой статье *множество слов* означает множество слов в некотором фиксированном большом конечном алфавите Σ. Этот алфавит достаточно большой для того, чтобы включить в себя символы всех описанных здесь множеств. Все машины Тьюринга рассматриваются как детерминированные распознающие устройства, если специально не оговаривается противное.

1. ТАВТОЛОГИИ И ПОЛИНОМИАЛЬНАЯ СВОДИМОСТЬ

Зафиксируем некоторый формализм для пропозиционного исчисления, в котором формулы записываются как слова в алфавите Σ. Так как нам потребуется бесконечное число пропозиционных символов (атомов), каждый такой символ будет состоять из некоторого элемента Σ, за которым следует число в двоичной записи, чтобы отличить этот символ от других. Таким образом, формула длины n может иметь только около $n/\log n$ различных функциональных и предикатных символов. Логическими связками являются & (и), $\vee$ (или), $\neg$ (не).

[1]) Cook S. A., The complexity of theorem-proving procedure, Proc. 3d Annual ACM Symposium on the Theory of Computing, Shaker Heights, Ohio, 1971, p. 151—159.

Множество тавтологий (обозначаемое через {тавтологии}) — это некоторое рекурсивное множество слов в этом алфавите, и мы интересуемся проблемой нахождения хорошей нижней границы для времени его возможного распознавания. Мы не даем здесь такой нижней границы, но теорема 1 будет свидетельствовать о том, что множество {тавтологии} является трудным для распознавания, так как многие кажущиеся трудными проблемы могут быть сведены к определению тавтологичности. Под *сводимостью* мы понимаем, грубо говоря, следующее: если бы тавтологичность решалась немедленно (с помощью «оракула»), то эти проблемы можно было бы решить за полиномиальное время. Для уточнения этого понятия мы введем спрашивающие машины, которые подобны машинам Тьюринга с оракулом [1].

Спрашивающей машиной называется многоленточная машина Тьюринга с одной выделенной лентой, называемой *лентой вопросов*, и тремя выделенными состояниями, называемыми соответственно *спрашивающим состоянием*, *да-состоянием* и *нет-состоянием*. Если M — спрашивающая машина и T — некоторое множество слов, то T-вычислением машины M называется такое вычисление машины M, при котором M в начале вычисления находится в своем начальном состоянии и входное слово w записано на ее входной ленте; и каждый раз, когда M попадает в спрашивающее состояние и на ленте вопросов записано слово u, то следующим состоянием M будет да-состояние, если $u \in T$, и нет-состояние, если $u \notin T$. Мы можем понимать под «оракулом» того, которому известно T и который переводит M в да-состояние или нет-состояние.

Определение

Множество слов S называется *P-сводимым* (P означает «полиномиально») к множеству слов T, если существует некоторая спрашивающая машина M и полином $Q(n)$, такие, что для каждого входного слова w T-вычисление на машине M со входом w заканчивается в течение $Q(|w|)$ шагов ($|w|$ — длина слова w) и заключительное состояние является благоприятным тогда и только тогда, когда $w \in S$.

Нетрудно видеть, что отношение P-сводимости является транзитивным. Следовательно, отношение E на множествах слов, задаваемое так: $(S, T) \in E$ тогда и только тогда, когда S и T P-сводимы друг к другу, является отношением эквивалентности. Класс эквивалентности, содержащий множество S, будет обозначаться через $\deg(S)$ (полиномиальная степень трудности S).

Определение. Обозначим $\deg(\{0\})$ через $\mathscr{L}_*$; через 0 здесь обозначена нулевая функция.

Таким образом, $\mathscr{L}_*$ — это класс множеств, распознаваемых за полиномиальное время. Класс $\mathscr{L}_*$ обсуждался в [2], стр. 5. Он является словарным аналогом кобхэмового класса функций [3].

Определим теперь следующие специальные множества слов.

1) *Проблема подграфа* заключается в том, что для двух данных конечных неориентированных графов требуется определить, является ли первый изоморфным некоторому подграфу второго. Граф G можно представить в виде слова $\tilde{G}$ в алфавите {0, 1, *}, выписывая последовательно строчки его матрицы смежности, отделенные друг от друга символами *. Пусть {пара подграфов} обозначает множество слов $\tilde{G}_1{**}\tilde{G}_2$, таких, что G_1 изоморфен подграфу G_2.

2) *Проблема изоморфизма графов* будет представляться множеством, обозначаемым {пара изоморфных графов}, всех слов $\tilde{G}_1{**}\tilde{G}_2$, таких, что G_1 изоморфен G_2.

3) Множество {простые числа} — это множество всех двоичных записей простых чисел.

4) Множество {ДНФ-тавтологии} — это множество слов, представляющих тавтологии в дизъюнктивной нормальной форме.

5) Множество D_3 состоит из всех тех тавтологий в дизъюнктивной нормальной форме, в которых каждая конъюнкция содержит не более трех членов (каждый из которых — атом или отрицание атома).

Теорема 1. *Если множество слов S распознаваемо некоторой недетерминированной машиной Тьюринга за полиномиальное время, то S P-сводимо к {ДНФ-тавтологии}.*

Следствие. *Каждое из множеств, определенных свойствами* 1—5, *P-сводится к {ДНФ-тавтологии}.*

Это имеет место потому, что каждое множество или его дополнение распознается за полиномиальное время на недетерминированной машине Тьюринга.

Доказательство теоремы

Предположим, что недетерминированная машина Тьюринга M распознает множество слов S за время $Q(n)$, где $Q(n)$ — полином. Для произвольного данного входа w машины M мы построим пропозициональную формулу $A(w)$ в конъюнктивной нормальной форме, такую, что $A(w)$ выполнима тогда и только тогда, когда M принимает w. Формула $A(w)$ легко переводится в дизъюнктивную нормальную форму (используя законы де Моргана), и полученная д. н. ф. является тавтологией тогда и только тогда, когда $w \notin S$. Так как вся эта конструкция может быть выполнена за время, ограниченное некоторым полиномом от $|w|$ (длины w), то теорема будет доказана. Кроме того, мы

можем предположить, что машина Тьюринга M имеет только одну ленту, бесконечную вправо и имеющую самую левую ячейку. Занумеруем ячейки слева направо числами 1, 2, 3, ... Зафиксируем вход w длины n, и пусть $w \in S$. Тогда существует вычисление машины M на входе w, которое оканчивается менее чем за $T = Q(n)$ шагов в допускающем состоянии. Формула $A(w)$ будет построена из многих различных пропозициональных символов, подразумеваемый смысл которых, приведенный ниже, относится к такому вычислению.

Пусть алфавит ленты машины M есть $\{\sigma_1, \ldots, \sigma_l\}$ и множество ее состояний $\{q_1, \ldots, q_r\}$. Отметим, что, так как вычисление имеет не более $T = Q(n)$ шагов, никакая ячейка с номером, большим T, не может обозреваться головкой.

Пропозициональные символы

$P^i_{s,t}$, где $1 \leqslant i \leqslant l$; $1 \leqslant s, t \leqslant T$. $P^i_{s,t}$ истинно тогда и только тогда, когда ячейка с номером s на шаге t содержит символ σ_i.

Q^i_t, где $1 \leqslant i \leqslant r$; $1 \leqslant t \leqslant T$. Q^i_t истинно тогда и только тогда, когда на шаге t машина находится в состоянии q_i.

$S_{s,t}$, где $1 \leqslant s, t \leqslant T$, истинно тогда и только тогда, когда на шаге t ячейка с номером s обозревается головкой.

Формула $A(w)$ является конъюнкцией $B \& C \& D \& E \& F \& G \& H \& I$, образованной следующим образом. Отметим, что $A(w)$ является конъюнктивной нормальной формой.

B будет утверждать, что на каждом шаге t обозревается одна и только одна ячейка. B является конъюнкцией $B_1 \& B_2 \& \ldots \& B_T$, где B_t утверждает, что на шаге t обозревается одна и только одна ячейка

$$B_t = (S_{1,t} \vee S_{2,t} \vee \ldots \vee S_{T,t}) \& \left[\&_{1 \leqslant i < j \leqslant T} (\neg S_{i,t} \vee \neg S_{j,t})\right].$$

Для $1 \leqslant s \leqslant T$ и $1 \leqslant t \leqslant T$ $C_{s,t}$ утверждает, что на шаге t в ячейке s находится один и только один символ. C является конъюнкцией всех таких $C_{s,t}$.

D утверждает, что для каждого t имеется одно и только одно состояние.

E утверждает, что выполнены начальные условия

$$E = Q^1_1 \& S_{1,1} \& P^{i_1}_{1,1} \& P^{i_2}_{2,1} \& \ldots \& P^{i_n}_{n,1} \& P^1_{n+1,1} \& \ldots \& P^1_{T,1},$$

где $w = \sigma_{i_1}\sigma_{i_2} \ldots \sigma_{i_n}$, q_1 — начальное состояние и σ_b — пустой символ.

F, G и H утверждают, что на каждом шаге t значения P, Q и S определены правильно. Например, G — это конъюнкция $G^t_{i,j}$ по всем t, i, j, где $G^t_{i,j}$ утверждает, что если на шаге t машина

находится в состоянии q_i и наблюдает символ σ_j, то на шаге $t+1$ машина будет в одном из состояний $q_{k_1}, q_{k_2}, \ldots, q_{k_m}$, где q_{k_i} задается функцией переходов машины M [1]).

$$G^t_{i,j} = \mathop{\&}_{s=1}^{T-1} (\neg Q^i_t \vee \neg S_{s,t} \vee \neg P^j_{s,t} \vee Q^{k_1}_{t+1} \vee Q^{k_2}_{t+1} \vee \ldots \vee Q^{k_m}_{t+1}).$$

Наконец, формула I утверждает, что на некотором шаге машина достигает благоприятного состояния. Машину M следует модифицировать так, чтобы после достижения благоприятного состояния она продолжала вычисления некоторым тривиальным образом, так что $A(w)$ будет выполнимой.

Теперь можно прямо проверить, что $A(w)$ обладает всеми свойствами, о которых утверждалось в первом параграфе доказательства.

Теорема 2. *Следующие множества попарно P-сводимы друг к другу* (*и, следовательно, имеют одну и ту же полиномиальную степень трудности*): {*тавтологии*}, {*ДНФ-тавтологии*}, D_3, {*пара подграфов*}.

Замечание. Мы не в состоянии добавить к приведенному выше списку ни {простые числа}, ни {пара изоморфных графов}. Для того, чтобы показать, что {тавтологии} P-сводится к {простые числа}, по-видимому, требуются некоторые глубокие результаты из теории чисел, в то время как доказательство того, что {тавтологии} P-сводится к {пара изоморфных графов}, вероятно, послужит опровержению одной гипотезы Корнеля [4], из которой он выводит, что проблема изоморфизма графов может быть решена за полиномиальное время.

Между прочим, из процедуры Девиса — Патнама [5] нетрудно видеть, что множество D_2, состоящее из всех ДНФ-тавтологий, в которых каждая конъюнкция имеет не более двух атомов, принадлежит $\mathscr{L}_*$. Следовательно, D_2 не может быть добавлено к списку множеств теоремы 2 (если только все множества этого списка не принадлежат $\mathscr{L}_*$).

Доказательство теоремы 2. Согласно следствию теоремы 1, каждое из указанных множеств P-сводится к {ДНФ-тавтологии}. Так как очевидно, что {ДНФ-тавтологии} P-сводится к {тавтологии}, остается показать, что {ДНФ-тавтологии} P-сводится к D_3 и D_3 P-сводится к {пара подграфов}.

---

[1]) Необходимо еще позаботиться о том, чтобы значения $S_{s,t+1}$, определяемые в формуле H, были согласованы (в смысле программы машины M) со значениями $Q^{k_i}_{t+1}$, определяемыми в формуле G. — *Прим. ред.*

Для того чтобы показать, что {ДНФ-тавтологии} P-сводится к D_3, предположим, что A — пропозициональная формула в дизъюнктивной нормальной форме. Скажем, $A = B_1 \vee \ldots \vee B_k$, где $B_1 = R_1 \& \ldots \& R_s$ и каждое R_i есть атом или отрицание атома и $s > 3$. Тогда A есть тавтология тогда и только тогда, когда A' есть тавтология, где $A' = P \& R_3 \& \ldots \& R_s \vee \neg P \& R_1 \& R_2 \vee B_2 \vee \ldots \vee B_k$ и P — новый атом. Таким образом, мы уменьшили число конъюнкций в B_1, и этот процесс можно повторять до тех пор, пока в конце концов не найдем формулы, которая имеет не больше трех атомов в каждой конъюнкции. Ясно, что весь процесс ограничен по времени некоторым полиномом от длины A.

Остается показать, что D_3 P-сводится к {пара подграфов}. Пусть A — формула в дизъюнктивной нормальной форме, каждая конъюнкция которой имеет длину три, т. е. $A = C_1 \vee C_2 \vee \ldots \vee C_k$, где $C_i = R_{i1} \& R_{i2} \& R_{i3}$ и каждое R_{ij} — атом или отрицание атома. Пусть теперь G_1 — полный граф с вершинами $\{v_1, \ldots v_k\}$ и G_2 — граф с вершинами $\{u_{i,j}\}$, $1 \leqslant i \leqslant k$; $1 \leqslant j \leqslant 3$, такой, что u_{ij} соединена дугой с u_{rs} тогда и только тогда, когда $i \neq r$ и две буквы (R_{ij}, R_{rs}) не образуют противоположной пары букв (т. е. они не имеют вида $(P, \neg P)$ или $(\neg P, P)$). Таким образом, для формулы A можно подобрать значения, при которых она становится ложной тогда и только тогда, когда существует гомоморфизм [1]) ϕ: $G_1 \to G_2$, такой, что для каждого i $\phi(v_i) = u_{ij}$ для некоторого j. (Гомоморфизм указывает для каждого i, который из атомов R_{i1}, R_{i2}, R_{i3} следует считать ложным, а выбор дуг в G_2 гарантирует, что истинностные значения для всех атомов совместимы.)

2. ОБСУЖДЕНИЕ

Теорема 1 и ее следствие являются сильным аргументом в пользу того, что не так-то просто определить, является ли данная пропозиционная формула тавтологией, даже если эта формула выражена в дизъюнктивной нормальной форме. Теоремы 1 и 2 вместе указывают на то, что поиск полиномиальной разрешающей процедуры для проблемы подграфа является делом неплодотворным, так как успех в этом деле сразу же дал бы полиномиальные разрешающие процедуры для многих других, по-видимому трудноразрешимых, проблем. Разумеется, подобное замечание относится к любой комбинаторной проблеме, к которой P-сводится {тавтологии}.

Более того, эти же теоремы свидетельствуют о том, что множество {тавтологии} является хорошим кандидатом для интересного множества, которое не принадлежит $\mathscr{L}_*$, и мне кажется, стоит потратить значительные усилия, стараясь доказать эту

[1]) Здесь имеется в виду изоморфное вложение G_1 в G_2. — *Прим. ред.*

гипотезу. Такое доказательство было бы большим успехом в теории сложности.

Принимая во внимание очевидную сложность множества {ДНФ-тавтологии}, интересно исследовать процедуру Девиса — Патнама [5]. Эта процедура была разработана для того, чтобы проверить, когда данная формула в конъюнктивной нормальной форме выполнима, но, очевидно, «двойственная» процедура определяет, является ли данная формула в дизъюнктивной нормальной форме тавтологией. Я еще не в состоянии найти серию примеров, показывающих, что эта процедура (производимая с осторожностью во избежание ловушек) должна требовать более чем полиномиального времени. Я также не нашел интересной верхней границы для требуемого времени.

Если мы будем рассматривать слова как записи натуральных чисел (или k-наборов натуральных чисел), используя m-ичную или другую подходящую запись, тогда понятия предыдущего пункта можно отнести к множествам чисел (или к k-местным отношениям чисел). Нетрудно видеть, что множество отношений, распознаваемых за полиномиальное время некоторой недетерминированной машиной Тьюринга, является в точности множеством $\mathscr{L}^+$ отношений вида

$$\exists y \leqslant g_k(\vec{x})\, R(\vec{x}, y), \tag{1}$$

где $g_k(\vec{x}) = 2^{(l(\max \vec{x}))^k}$, $l(z)$ — двоичная длина z и $R(\vec{x}, y)$ — некоторое отношение из $\mathscr{L}_*$ ($\mathscr{L}^+$ — класс расширенных положительных рудиментарных отношений Беннета [6]). Если бы мы убрали границу квантора в формуле (1), то класс $\mathscr{L}^+$ превратился бы в класс всех рекурсивно перечислимых множеств. Таким образом, если $\mathscr{L}^+$ рассматривать как аналог класса рекурсивно перечислимых множеств, то определение тавтологичности будет аналогом проблемы останова; следовательно, согласно теореме 1, тавтологии имеют полную степень точно так же, как проблема останова имеет полную рекурсивно перечислимую степень. К сожалению, диагональная процедура, которая показывает, что проблема останова не рекурсивна, по-видимому, не может быть приспособлена для того, чтобы показать, что {тавтологии} не принадлежат $\mathscr{L}_*$.

3. ИСЧИСЛЕНИЕ ПРЕДИКАТОВ

Формулы исчисления предикатов представляются словами подобно формулам пропозиционального исчисления. В дополнение к символам для пропозициональных букв и логических связок нам необходимы кванторные символы $\forall$ и $\exists$ и символы

для формирования бесконечного списка индивидных переменных и бесконечных списков функциональных и предикатных символов любого порядка (конечно, полагая, что алфавит Σ конечный).

Пусть Q — некоторая процедура, оперирующая с такими формулами, которая заканчивается для данной входной формулы A тогда и только тогда, когда A невыполнима. Так как не существует разрешающей процедуры для распознавания выполнимости в исчислении предикатов, то отсюда следует, что не существует рекурсивной функции T, такой, что если A невыполнима, то Q заканчивается в течение $T(n)$ шагов, где n — длина A. Как же тогда оценивать эффективность процедуры?

Мы применим следующий подход. Большинство из тех, кто занимается автоматическим выводом теорем, исходят из теоремы Эрбрана, которая, коротко говоря, утверждает, что формула A невыполнима тогда и только тогда, когда некоторая конъюнкция подстановковых частных случаев функциональной формы $\mathrm{fn}(A)$ формулы A по истинности функционально несовместна. Предположим, что мы упорядочили термы универсума Эрбрана для $\mathrm{fn}(A)$ по рангам и затем упорядочили некоторым естественным образом подстановковые частные случаи $\mathrm{fn}(A)$ из универсума Эрбрана. Это упорядочивание должно быть таким, чтобы, вообще говоря, подстановковые частные случаи, которые используют термы большего ранга, следовали за подстановковыми частными случаями, использующими термы меньшего ранга. Пусть $A_1, A_2, \ldots$ — это подстановковые частные случаи, упорядоченные таким образом.

Определение. Если A невыполнима, то пусть $\phi(A)$ — такое наименьшее k, что конъюнкция $A_1 \& A_2 \& \ldots \& A_k$ по истинности функционально несовместна. Если A выполнима, то $\phi(A)$ не определено.

Пусть теперь Q — процедура, которая по данному A вычисляет последовательность $A_1, A_2, \ldots$ и для каждого i проверяет, верно ли, что $A_1 \& \ldots \& A_i$ по истинности функционально совместна. Процедура заканчивается успехом, если ответ будет «нет». Тогда, очевидно, существует такая рекурсивная функция $T(k)$, что для всех k и всех формул A, если длина $A \leqslant k$ и $\phi(A) \leqslant k$, то Q оканчивается за $T(k)$ шагов. Мы полагаем, что функция $T(k)$ есть мера эффективности Q.

Для удобства все процедуры в этом параграфе будут реализоваться на одноленточных машинах Тьюринга, которые мы будем называть просто *машинами*.

Определение. Для данных машины M_Q и рекурсивной функции $T_Q(k)$ мы будем говорить, что M_Q *имеет тип* Q *и работает в течение времени* $T_Q(k)$ при условии, что когда M_Q начи-

нает работу с предикатной формулы A, записанной на ее ленте, то M_Q останавливается тогда и только тогда, когда A невыполнима, и для всех k, если $\phi(A) \leqslant k$ и $|A| \leqslant \log_2 k$, то M_Q останавливается за $T_Q(k)$ шагов. В этом случае мы также будем говорить, что $T_Q(k)$ имеет тип Q. Здесь $|A|$ — длина A.

Мы взяли $|A| \leqslant \log_2 k$ вместо $|A| \leqslant k$ по той причине, что для последнего условия нахождение нижней границы для $T_Q(k)$ было бы примерно эквивалентным нахождению нижней границы для проблемы разрешения пропозиционального исчисления. В частности, теорема 3А стала бы очевидной и тривиальной.

Теорема 3. А) *Для любой $T_Q(k)$ типа Q отношение*

$$\frac{T_Q(k)}{\sqrt{k}/(\log k)^2} \tag{2}$$

неограничено.

В) *Существует такая $T_Q(k)$ типа Q, что*

$$T_Q(k) \leqslant k \cdot 2^{k\,(\log k)^2}.$$

Набросок доказательства. А) По данной машине M можно построить предикатную формулу $A(M)$, которая выполнима тогда и только тогда, когда M, начиная работу с пустой лентой, не останавливается. Эта конструкция описана Ваном [7] в доказательстве, сводящем проблему останова к проблеме разрешения исчисления предикатов. Более того, если M останавливается за s шагов, то $\phi(A(M)) \leqslant s^2$. Таким образом, если, в противоречие с (2) $T_Q(k) = O\left(\sqrt{k}/\log^2 k\right)$, то, модифицируя M_Q, можно было бы проверить только за $O\left(\sqrt{s^2}/\log^2 s^2\right)$ шагов, что M остановится за s шагов (при условии, что $m \leqslant \log s^2$, где m — длина $A(M)$). Диагональный метод (см. [8], стр. 153) показывает, что это в общем случае невозможно.

В) Машина M_Q работает в течение времени T_Q, следуя процедуре, изложенной в начале этого параграфа. Заметим, что формула $A_1 \,\&\, A_2 \,\&\, \ldots \,\&\, A_k$ имеет длину $O(k \log^2 k)$, так как мы можем предполагать, что $|A| \leqslant \log_2 k$.

Теорема 4. *Если множество S слов распознаваемо недетерминированной машиной за время $T(n) = 2^n$ и если $T_Q(k)$ является честной (т. е. вычислимой в реальное время) функцией типа Q, то существует константа k, такая, что S может быть распознано детерминированной машиной за время $T_Q(k \cdot 16^n)$.*

Доказательство. Пусть M_1 — недетерминированная машина, распознающая S за время 2^n. Пусть M_2 — недетерминированная машина, которая моделирует M_1 в течение точно 2^n шагов и затем останавливается, если M_1 не принимает данного

входного слова. В противном случае M_2 вычисляет вечно. Таким образом, для всех слов w, если $w \in S$, то существует вычисление, при котором M_2 со входом w не останавливается, и если $w \notin S$, то M_2 со входом w останавливается в пределах 4^n шагов для *всех вычислений*. Взяв w длины n, мы можем построить формулу $A(w)$ длины $O(n)$, такую, что $A(w)$ выполнима тогда и только тогда, когда M_1 принимает w. ($A(w)$ строится подобно тому, как строится $A(w)$ в доказательстве теоремы 1.) Более того, если M_2 останавливается за 4^n шагов для всех возможных вычислений, то $\phi(A(w)) \leqslant k(4^n)^2 = k\,16^n$. Таким образом, можно построить детерминированную машину M для определения того, что $w \in S$, представляя M_Q со входом $A(w)$. Если результат не появится за $T_Q(k\,16^n)$ шагов, то $w \in S$, в противном случае $w \notin A$.

4. ДАЛЬНЕЙШИЕ ОБСУЖДЕНИЯ

Существует большой разрыв между нижней границей $\sqrt{k}/(\log k)^2$ для временной функции $T_Q(k)$, которую дает теорема 3А, и возможной границей $T_Q(k) = k2^{k(\log k)^2}$, которая дается в 3В. Однако имеются основания для такого разрыва. Например, если бы мы могли улучшить результат в 3В и найти $T_Q(k)$, ограниченную некоторым полиномом от k, то по теореме 4 мы могли бы моделировать недетерминированную машину, ограниченную временем 2^n, недетерминированной машиной за $P(2^n)$ шагов, где P — некоторый полином. Это противоречит опыту, который показывает, что для детерминированного моделирования $T(n)$ шагов работы недетерминированной машины требуется, вообще говоря, $k^{T(n)}$ шагов.

С другой стороны, если бы мы могли повысить нижнюю оценку, данную теоремой 3А, и показать, что $T_Q(k)/2^k$ неограничено, то мы могли бы сделать заключение, что {тавтологии} $\notin \mathscr{L}_*$, так как в противном случае общая процедура Эрбрана требовала бы $T_Q(k)$, меньшего, чем 2^k. Следовательно, такое улучшение в 3А потребовало бы решения главного вопроса в теории сложности.

Область механического доказательства теорем сильно нуждается в основе для сравнения и оценки десятков различных процедур, которые появляются в литературе. Выполнение этих процедур на вычислительных машинах является хорошим критерием, но не достаточным (если только данная процедура не является полезной в некотором практическом смысле). Необходимы теоретические критерии сложности, которые дадут нам фундаментальные ограничения и выдвинут новые цели. Критерий, предложенный здесь (функция $T_Q(k)$), является, вероятно, слишком грубым. Например, быть может, лучше сделать $T_Q(k)$

функцией от нескольких переменных, одна из которых — $\phi(A)$, а другая может быть минимальным числом подстановковых частных случаев $\mathrm{fn}(A)$, необходимых для того, чтобы получить противоречие (заметим, что для этого нужны, вообще говоря, не все $A_1, \ldots, A\phi(A)$).

$T_Q(k)$ может быть грубой мерой, но она действительно дает некоторую основу для обсуждения и, я надеюсь, будет стимулировать прогресс на пути к нахождению лучших мер сложности для доказательства теорем.

ЛИТЕРАТУРА

1. Kreider K. L., Ritchie R. W., Predictably computable functionals and definitions by recursion, *Zeitschrift fur Math. Logik und Grundlagen der Math.*, **10**, (1964) 65—80.
2. Cook S. A., Charakcterizations of pushdown machines in terms of time-bounded computers, *Journ. Assoc. Comp. Mach.*, **18**, № 1, (1971), 4—8. (Русский перевод: Кук С. Характеристики магазинных автоматов в терминах вычислителей, ограниченных во времени, в сб. пер. «Сложность вычислений и алгоритмов», М., 1974, 266—288.)
3. Cobham. Alan, The intrinsic computational difficulty of functions, Proc. of the International Congress for logic, methedology and the philosophy of science, North Holland Publishing Co., Amsterdam, pp. 24—30.
4. Corneil D. G., Gotleb C. C., An efficient algorithm for graph isomorphism, *Journ. Assoc. Comp. Mach.*, **17**, № 1, (1970), 51—64.
5. Davis M., Putnam H., A computing procedure for quantification theory, *Journ. Assoc. Comp. Mach.* (1960) 201—215.
6. Bennett J. H., On spectra, Doctoral dissertation, Princeton University, 1962.
7. Hao Wang, Dominoes and the $\forall\exists\forall$ case of the decision problem, Proc. of the symposium on mathematical theory of automata at Polytechnic Institute of Brooklin, 1962, pp. 23—55.
8. Hopcroft J., Ullman J., Formal languages and their relation to automata, Addison-Wesley, 1969.

Сводимость комбинаторных проблем[1])

Ричард М. Карп

Калифорнийский университет, Беркли

Большой класс вычислительных проблем включает определение свойств графов, ориентированных графов, целых чисел, массивов целых чисел, конечных семейств конечных множеств, булевых формул и элементов других счетных множеств. Посредством простого кодирования таких объектов множествами слов над конечным алфавитом эти проблемы могут быть превращены в проблемы распознавания языков и можно исследовать их вычислительную сложность. Имеет смысл считать, что такая проблема решается удовлетворительно, если найден алгоритм для ее решения, оканчивающий свою работу за время (число шагов), ограниченное полиномом от длины входного слова. Мы покажем, что большое число классических нерешенных проблем покрытия, сочетания, упаковки, нахождения маршрутов, назначения и упорядочения эквивалентны между собой в том смысле, что или каждая из них обладает полиномиально-ограниченным алгоритмом, или ни одна из них такого алгоритма не имеет.

1. ВВЕДЕНИЕ

Все известные в настоящее время общие методы вычисления хроматического числа графа, определения существования гамильтонова цикла в графе или решения системы линейных неравенств, в которых переменные принимают значения 0 или 1, требуют вычислений, длительность которых растет в худшем случае экспоненциально по отношению к длине слова. В настоящей статье мы докажем ряд теорем, которые дают повод (хотя прямо из них это не следует) считать, что эти проблемы, а также многие другие будут всегда оставаться трудными для решения.

Нас особенно интересует существование алгоритмов, обеспечивающих завершение работы за число шагов, ограниченное полиномом от длины входа. Мы укажем класс хорошо известных комбинаторных проблем, включая упомянутые выше, которые эквивалентны друг другу в том смысле, что полиномиально-огра-

[1]) Karp R. M., Reducibility among Combinatorial problems, Complexity of computer computations. Proc. Symp. March 20—22, 1972, 85—103.

ниченный алгоритм для любой из них будет эффективно порождать полиномиально-ограниченные алгоритмы для всех. Мы также покажем, что если эти проблемы обладают полиномиально-ограниченными алгоритмами, то все проблемы из очень широкого класса проблем (грубо говоря, класса проблем, разрешимых путем обратного поиска полиномиальной глубины) обладают полиномиально-ограниченными алгоритмами.

Изложим вкратце содержание настоящей статьи. Ради определенности изложение ведется в терминах распознавания языков на одноленточных машинах Тьюринга, но любая другая модель из довольно широкого множества абстрактных моделей вычислений позволяет развить подобную теорию. Пусть Σ^* — множество всех конечных слов, составленных из нулей и единиц. Произвольное подмножество Σ^* называется *языком*. Пусть P — класс языков, распознаваемых за полиномиальное время на одноленточных детерминированных машинах Тьюринга, и пусть NP — класс языков, распознаваемых за полиномиальное время на одноленточных недетерминированных машинах Тьюринга. Пусть Π — класс функций, определенных на Σ^* со значениями из Σ^*, вычислимых за полиномиальное время на одноленточных машинах Тьюринга. Пусть L и M — языки. Говорят, что $L \propto M$ (L *сводится к* M), если существует функция $f \in \Pi$, такая, что $f(x) \in M \Leftrightarrow x \in L$. Если $M \in P$ и $L \propto M$, то $L \in P$. Назовем L и M эквивалентными, если $L \propto M$ и $M \propto E$. Назовем L (*полиномиально*) *полным*, если $L \in NP$ и любой язык из NP сводится к L. Очевидно, что либо все полные языки принадлежат P, либо ни один из них не принадлежит P. Первое имеет место тогда и только тогда, когда $P = NP$.

Главное содержание этой статьи заключается в доказательстве того, что большое число классических трудных вычислительных проблем, возникающих в таких областях, как математическое программирование, теория графов, комбинаторика, вычислительная логика, теория переключательных схем, являются полными (и, следовательно, эквивалентными), когда они выражаются естественным путем как проблемы распознавания языков.

Эта статья навеяна работой Стефана Кука [1] и опирается на важную теорему из этой работы. Автор признателен Юджину Лоулеру и Роберту Тарьяну за большую помощь.

2. КЛАСС P

Существует большой класс важных вычислительных проблем, включающий в себя определение свойств графов, ориентированных графов, целых чисел, конечных семейств конечных множеств, булевых формул и элементов других счетных областей.

Разумная рабочая гипотеза, которую впервые стал защищать Джек Эдмондс [2] в связи с проблемами теории графов и целочисленного программирования и которая поныне широко принята, состоит в том, что такого рода проблема может считаться легко решаемой тогда и только тогда, когда существует алгоритм ее решения со временем работы, которое ограничено полиномом от размера входных данных. В этом разделе мы вводим и начинаем исследовать класс проблем, разрешимых за полиномиальное время.

Начнем с того, что дадим предельно общее определение понятия «детерминированного алгоритма», вычисляющего функцию, определенную на счетном множестве D, принимающую значение из счетного множества R.

Для любого конечного алфавита A обозначим через A^* множество конечных слов алфавита A, для любого $x \in A^*$ пусть $\lg(x)$ обозначает длину x.

Детерминированный алгоритм A характеризуется следующим:

счетным множеством D (область определения),
счетным множеством R (область значений),
конечным алфавитом Δ, таким, что $\Delta^* \cap R = \varnothing$,
кодирующей функцией E: $D \to \Delta^*$,
функцией перехода τ: $\Delta^* \to \Delta^* \cup R^*$ [1])

Вычисление A на входе $x \in D$ есть единственная последовательность $y_1, y_2, \ldots$, такая, что $y_1 = E(x)$, $y_{i+1} = \tau(y_i)$ для любого i, и если эта последовательность конечна и заканчивается y_k, то $y_k \in R$. Любое слово, являющееся элементом вычисления, называется *мгновенным описанием*. Если вычисление A на входе x является конечной последовательностью длины $t(x)$, то *время работы* A на входе x равно $t(x)$. A называется ограниченным, если все его вычисления конечны. Ограниченный алгоритм A вычисляет функцию f_A: $D \to R$, такую, что $f_A(x)$ есть последний элемент вычисления A на x.

Если $R = \{$ПРИНЯТЬ, ОТВЕРГНУТЬ$\}$, то A называется *распознающим алгоритмом*. Распознающий алгоритм, для которого $D = \Sigma^*$, *называется словарным распознающим алгоритмом*. Если A — словарный распознающий алгоритм, то *язык, распознаваемый* алгоритмом A, — это $\{x \in \Sigma^* | f_A(x) = \text{ПРИНЯТЬ}\}$. Если $D = R = \Sigma^*$, то A называется *словарным отображающим алгоритмом*. Ограниченный алгоритм A с областью определения $D = \Sigma^*$ работает *полиномиальное время*, если существует полином $p(\cdot)$, такой, что для любого $x \in \Sigma^*$, $t(x) \leqslant p(\lg(x))$.

Для того чтобы рассматривать алгоритмы в некотором прак-

[1]) Необходимо потребовать, чтобы кодирующая функция и функция перехода были вычислимыми.

тическом контексте, необходимо конкретизировать понятие детерминированного алгоритма. Различные хорошо известные классы словарных распознающих алгоритмов (алгоритмы Маркова, одноленточные машины Тьюринга, многоленточные и многоголовочные машины Тьюринга, машины со случайным доступом и т. д.) описываются путем наложения ограничений на функции E и τ так, чтобы они принадлежали определенным очень простым типам. Эти определения стандартны (Хопкрофт и Ульман [3]) и не будут здесь повторяться.

В настоящее время принято считать, что различные такие классы эквивалентны относительно возможности распознавания языков, для каждого такого класса алгоритмов класс распознаваемых языков является классом рекурсивных языков. Эта инвариантность относительно изменения определений — одно из свидетельств в пользу того, что рекурсивность является правильной математической формулировкой интуитивного понятия разрешимости.

Класс языков, распознаваемых словарными распознающими алгоритмами за полиномиальное время, также инвариантен относительно широкого ряда изменений в классе алгоритмов. Например, любой язык, распознаваемый за время $p(\cdot)$ на многоголовочной или многоленточной машине Тьюринга, распознаваем за время $p^2(\cdot)$ на одноленточной машине Тьюринга. Таким образом, класс языков, распознаваемых за полиномиальное время на одноленточной машине Тьюринга, тождествен классу, распознаваемому на, казалось бы, более мощных многоголовочных или многоленточных машинах Тьюринга. Подобные замечания применимы к машинам со случайным доступом.

Определение 1. P есть класс языков, распознаваемых на одноленточных машинах Тьюринга за полиномиальное время.

Определение 2. Π есть класс функций, определенных на Σ^* со значениями из Σ^*, вычисляемых на одноленточных машинах Тьюринга за полиномиальное время.

Читатель не допустит ошибки, если отождествит P с классом языков, распознаваемых на цифровых вычислительных машинах (с неограниченной периферийной памятью) за полиномиальное время, и Π — с классом словарных отображений, которые выполняются за полиномиальное время на таких вычислительных машинах.

Замечание. Если f: $\Sigma^* \to \Sigma^*$ принадлежит Π, то существует полином $p(\cdot)$, такой, что $\lg(f(x)) \leqslant p(\lg(x))$.

Теперь мы введем понятие сводимости, которое является важнейшим понятием этой статьи.

Определение 3. Пусть L и M — языки. Тогда $L \propto M$ (L сводим к M), если существует функция $f \in \Pi$, такая, что $f(x) \in M \Leftrightarrow x \in L$.

Лемма 1. *Если* $L \propto M$ *и* $M \in P$, *то* $L \in P$.

Доказательство. Полиномиально-ограниченный алгоритм, решающий $x \in L$, получается следующим образом: вычислить $f(x)$; затем проверить за полиномиальное время, выполняется ли $f(x) \in M$.

Нас будет интересовать трудность распознавания подмножеств счетных множеств, отличных от Σ^*. Если такое множество D зафиксировано, то обычно существует естественное взаимно однозначное кодирование e: $D \to \Sigma^*$. Например, мы можем представить положительное целое число как слово из нулей и единиц, являющееся двоичным представлением этого числа, одномерный целочисленный массив — как список целых чисел, матрицу — как список одномерных массивов и т. д. Существуют стандартные технические приемы кодирования «списков» словами над конечным алфавитом и слов над произвольным конечным алфавитом словами из нулей и единиц. Для данного кодирования e: $D \to \Sigma^*$ мы говорим, что множество $T \subseteq D$ распознается за полиномиальное время, если $e(T) \in P$. Пусть также даны множества $T \subseteq D$ и $U \subseteq D'$ и кодирующие функции e: $D \to \Sigma^*$ и e': $D' \to \Sigma^*$. Мы говорим $T \propto U$, если $e(T) \propto e'(U)$.

Как правило, возможны несколько естественных кодирований данного множества. Например, граф может быть представлен своей матрицей смежности, своей матрицей инцидентности или списком неупорядоченных пар вершин, соответствующих дугам. Если мы выбираем одно из этих представлений, то тем самым оказываются принятыми по нашему произволу несколько решений, касающихся, например, формата и пунктуации. К счастью, почти всегда «очевидно», что любые две разумные системы кодирования e_0 и e_1 данной проблемы эквивалентны, т. е. $e_0(S) \in P \Leftrightarrow e_1(S) \in P$. Одно важное исключение касается представления положительных целых чисел. Мы оговариваем, что положительное целое число кодируется в бинарной, а не в унарной форме. Исходя из инвариантности распознаваемости в полиномиальное время и сводимости друг к другу разумных кодирований, мы обсуждаем все проблемы в терминах их областей определения, не задавая их в кодах Σ^*.

Мы завершим этот раздел перечислением проблем, разрешимых в полиномиальное время. В следующем разделе мы рассматриваем несколько родственных по отношению к ним проблем, о которых неизвестно, разрешимы ли они в полиномиальное время. В приложении 1 приводится система обозначений.

Каждая проблема определяется заданием (под заголовком «ВХОД») элемента, характеризующего область определения, и (под заголовком «СВОЙСТВО») свойства, которое определяет, следует ли данный вход принять.

2-ВЫПОЛНИМОСТЬ (Выполнимость КНФ не более чем с двумя буквами в каждой скобке) [Кук [1]].

ВХОД: Скобки (элементарные дизъюнкции) $C_1, C_2, \ldots, C_p$, каждая из которых содержит не более двух букв

СВОЙСТВО: Конъюнкция данных скобок выполнима, т. е. существует множество $S \subseteq \{x_1, x_2, \ldots, x_n; \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$, такое, что

а) S не содержит пары букв, одно из которых является отрицанием другой, и

б) $S \cap C_k \neq \varnothing$, $k = 1, 2, \ldots, p$.

МИНИМАЛЬНЫЙ ОСТОВ (накрывающее дерево) графа [Крускал [5]]

ВХОД: G, w, W

СВОЙСТВО: Существует остов веса $\leqslant W$.

КРАТЧАЙШИЙ ПУТЬ [Дийкстра [6]]

ВХОД: G, w, W, s, t

СВОЙСТВО: Существует путь между s и t веса $\leqslant W$.

МИНИМАЛЬНЫЙ РАЗРЕЗ [Эдмондс и Карп [7]]

ВХОД: G, w, W, s, t

СВОЙСТВО: Существует s, t разрез веса $\leqslant W$.

РЕБЕРНОЕ ПОКРЫТИЕ [Эдмондс [2]]

ВХОД: G, k

СВОЙСТВО: Существует множество $Y \subseteq A$, такое, что $|Y| \leqslant k$ и каждая вершина инцидентна некоторому ребру в Y.

УДАЛЕНИЕ РЕБЕР

ВХОД: G, k

СВОЙСТВО: Существует множество из k ребер, удаление которых разрывает все циклы.

ДВУДОЛЬНОЕ ПАРОСОЧЕТАНИЕ [Холл [8]]

ВХОД: $S \subseteq Z_p \times Z_p$

СВОЙСТВО: Существуют p элементов S, любые два из которых различаются между собой в обоих компонентах.

УПОРЯДОЧЕНИЯ С ОГРАНИЧЕНИЯМИ

ВХОД: $(T_1, \ldots, T_n) \subset Z^n$, $(D_1, \ldots, D_n) \in Z^n$, k

СВОЙСТВО: Начиная со времени 0, можно в некоторой очередности выполнять работы 1, 2, 3, ..., n со временем выполнения T_i и сроками выполнения D_i, причем порядок должен быть таким, что не более чем для k работ нарушаются их сроки выполнения.

РАЗРЕШИМОСТЬ ЛИНЕЙНЫХ УРАВНЕНИЙ

ВХОД: (c_{ij}), (a_i)

СВОЙСТВО: Существует вектор (y_j), такой, что для каждого i $\sum_j c_{ij}y_j = a_i$.

3. НЕДЕТЕРМИНИРОВАННЫЕ АЛГОРИТМЫ И ТЕОРЕМА КУКА

В этом разделе мы сформулируем важную теорему, доказанную Куком [1] и утверждающую, что любой язык из определенного широкого класса языков NP сводится к конкретному языку S, соответствующему проблеме выполнимости булевой формулы, заданной в конъюнктивной нормальной форме.

Пусть $P^{(2)}$ обозначает класс подмножеств множества $\Sigma^* \times \Sigma^*$, распознаваемых за полиномиальное время. Для данных $L^{(2)} \in P^{(2)}$ и полинома p определим язык L следующим образом:

$L = \{x|$ существуют y, такие, что $\langle x, y\rangle \in L^{(2)}$ и $\lg(y) \leqslant p(\lg(x))\}$.

Будем говорить, что язык L выводим из $L^{(2)}$ путем навешивания p-ограниченного квантора существования.

Определение 4. NP есть множество языков, выводимых из элементов $P^{(2)}$ путем навешивания полиномиально-ограниченных кванторов существования.

Существует другая характеристика NP в терминах недетерминированных машин Тьюринга. *Недетерминированный распознающий алгоритм* A определяется:

> счетным множеством D (область *определения*),
> конечным алфавитом Δ, таким, что $\Delta \cap \{$ПРИНЯТЬ, ОТВЕРГНУТЬ$\} = \varnothing$,
> кодирующей функцией E: $D \to \Delta^*$,
> отношением перехода [1]) $\tau \subseteq \Delta^*(\Delta^* \cup \{$ПРИНЯТЬ, ОТВЕРГНУТЬ$\})$,

так, что для любого $y_0 \in \Delta^*$ множество $\{\langle y_0, y\rangle \,|\, \langle y_0, y\rangle \in \tau\}$ имеет меньше чем k_A элементов, где k_A — константа. *Вычисление* A на входе $x \in D$ есть последовательность $y_1, y_2, \ldots$, такая,

[1]) См. примечание на стр. 18.

что $y_1 = E(x)$, $\langle y_i, y_{i+1}\rangle \in \tau$ для всех i, и если последовательность конечна и оканчивается на y_k, то $y_k \in$ {ПРИНЯТЬ, ОТВЕРГНУТЬ}. Слово, встречающееся в некотором вычислении, называется *мгновенным описанием*. Конечное вычисление, оканчивающееся словом ПРИНЯТЬ, называется *принимающим вычислением*. Вход x называется *допустимым*, если для x существует принимающее вычисление. Если $D = \Sigma^*$, то A называется недетерминированным словарным распознающим алгоритмом и мы говорим, что A *работает полиномиальное время*, если существует полином $p(\cdot)$, такой, что если A принимает x, то для x существует принимающее вычисление длины $\leqslant p(\lg(x))$.

Недетерминированый алгоритм может рассматриваться как процесс, который, будучи поставлен перед выбором одной из (скажем) двух альтернатив, может создать две копии самого себя и проследить последовательность вычислений обоих способов действия. Такое ветвление может привести к экспоненциально увеличивающемуся числу копий. Вход является допустимым, если некоторая последовательность выборов приводит к его принятию.

Недетерминированные одноленточные машины Тьюринга, многоленточные машины Тьюринга, машины со случайным доступом и т. д. определяют классы недетерминированных словарных распознающих алгоритмов путем ограничения кодирующей функции F и отношения перехода τ до особенно простых форм. Все эти классы алгоритмов, ограниченных полиномиальным временем работы, определяют один и тот же класс языков. Более того, этот класс есть NP.

Теорема 1. *$L \in NP$ тогда и только тогда, когда L принимается*[1]) *недетерминированной машиной Тьюринга, работающей полиномиальное время.*

Доказательство. 1) Предположим, $L \in NP$. Тогда для некоторого $L^{(2)} \in P^{(2)}$ и некоторого полинома p язык L может быть получен из $L^{(2)}$ путем навешивания p-ограниченного квантора существования. Можно построить недетерминированную машину, которая первым делом «угадывает» последовательность цифр цепочки y длины $\leqslant p(\lg(x))$ и затем проверяет, будет ли $\langle x, y\rangle \in L^2$. Ясно, что такая машина распознает язык L за полиномиальное время.

2) Предположим, что язык L принимается недетерминированной машиной Тьюринга T за время p. Предположим, без ограничения общности, что для любого мгновенного описания Z

[1]) Язык L принимается машиной Тьюринга, если всякое слово из L принимается ею. — *Прим. ред.*

существует не более двух мгновенных описаний, которые могут следовать за Z (т. е. применимы не более чем два простых перехода). Тогда последовательность выборов мгновенных описаний, производимых T в данном вычислении, может быть закодирована словом y, состоящим из нулей и единиц, таким, что $\lg(y) \leqslant p(\lg(x))$.

Таким образом, можно построить детерминированную машину Тьюринга T' со входами из $\Sigma^* \times \Sigma^*$, которая на $\langle x, y\rangle$ действует подобно тому, как на входе x действует машина T с последовательностью выборов y. Ясно, что T' работает полиномиальное время и L получен при помощи полиномиально-ограниченного квантора существования из множества пар цепочек, принимаемых машиной T'.

Класс NP весьма широк. Допуская вольность речи, можно сказать, что данная проблема принадлежит классу NP тогда и только тогда, когда она может быть решена путем обратного поиска полиномиально-ограниченной глубины. Большое число важных вычислительных проблем, для которых неизвестно, принадлежат ли они P, очевидно, принадлежит NP. Например, проблема определения: могут ли вершины графа G быть раскрашены красками таким образом, что никакие две соседние вершины не имеют одинакового цвета. Недетерминированный алгоритм может просто наугад задать цвета вершин и затем проверить (за полиномиальное время), имеют ли все пары соседних вершин различный цвет.

Ввиду большой широты класса NP формируемая ниже и принадлежащая Куку теорема является весьма примечательной. Проблема выполнимости определяется следующим образом:

ВЫПОЛНИМОСТЬ

ВХОД: Скобки (элементарные дизъюнкции) $C_1, C_2, \ldots, C_p$.

СВОЙСТВО: Конъюнкция данных скобок является выполнимой, т. е. существует множество $S \subseteq \{x_1, x_2, \ldots, x_n; \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\}$, такое, что

а) S не содержит никакой буквы вместе с ее отрицанием и

б) $S \cap C_k = \varnothing$, $k = 1, 2, \ldots, p$.

Теорема 2. (Кук.) *Если* $L \in NP$, то $L \propto$ ВЫПОЛНИМОСТЬ.

Теорема, установленная Куком [1], использует более слабое понятие сводимости, чем то, которое используется здесь, но доказательство Кука справедливо и для нашего случая.

Следствие 1. $P = NP \Leftrightarrow$ ВЫПОЛНИМОСТЬ $\in P$.

Доказательство. Если ВЫПОЛНИМОСТЬ $\in P$, то для любого $L \in NP$, $L \in P$, так как $L \propto$ ВЫПОЛНИМОСТЬ. Если

ВЫПОЛНИМОСТЬ $\notin P$, то $P \neq NP$, так как ясно, что ВЫПОЛНИМОСТЬ $\in NP$.

Замечание. Если $P = NP$, то NP замкнут относительно дополнения и полиномиально ограниченного квантора существования. Следовательно, он также замкнут относительно полиномиально-ограниченного квантора общности. Из этого следует, что полиномиально-ограниченный аналог арифметической иерархии Клини (Роджерс [9]) становится тривиальным, если $P = NP$.

Теорема 2 показывает, что если бы существовал полиномиально ограниченный алгоритм распознавания ВЫПОЛНИМОСТИ, то любая проблема, разрешимая с помощью обратного поиска полиномиальной глубины, разрешима также с помощью (детерминированного) алгоритма за полиномиальное время. Это сильное косвенное свидетельство в пользу того, что ВЫПОЛНИМОСТЬ $\notin P$.

4. ПОЛНЫЕ ПРОБЛЕМЫ

Главная цель этой статьи — установление того, что большое число важных вычислительных проблем может играть роль проблемы ВЫПОЛНИМОСТЬ в теореме Кука. Такие проблемы мы будем называть полными.

Определение 5. Язык L является (полиномиально) *полным*, если

а) $L \in NP$.

б) ВЫПОЛНИМОСТЬ $\propto L$.

Теорема 3. *Либо все полные языки принадлежат P, либо ни один из них не принадлежит P. Первое имеет место тогда и только тогда, когда $P = NP$.*

Можно расширить понятие полноты, распространив его на проблемы, определенные над счетными множествами, отличными от Σ^*.

Определение 6. Пусть D — счетное множество, e — «стандартное» взаимно однозначное кодирование e: $D \to \Sigma^*$ и T — подмножество D. T называется полным тогда и только тогда, когда $e(T)$ полно.

Лемма 2. *Пусть D и D' — счетные множества с взаимно однозначными кодирующими функциями e и e'. Пусть $T \subseteq D$ и $T' \subseteq D'$. Тогда $T \propto T'$, если существует функция F: $D \to D'$, такая, что*

а) $F(x) \in T' \Leftrightarrow x \in T$ *и*

б) *существует функция* $f \in \Pi$, *такая, что* $f(x) = e'(F(e^{-1}(x)))$ *во всех случаях, когда* $e'(F(e^{-1}(x)))$ *определено.*

Оставшаяся часть статьи главным образом посвящена доказательству следующей теоремы.

Основная теорема. *Все нижеследующие проблемы полные.*

1. ВЫПОЛНИМОСТЬ

Примечание. В силу двойственности эта проблема эквивалентна определению того, является ли некоторая дизъюнктивная нормальная форма тавтологией.

2. 0-1 ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ
ВХОД: матрица целых чисел C и вектор целых чисел d
СВОЙСТВО: Существует вектор x из нулей и единиц (0-1 вектор), для которого $Cx = d$.

3. КЛИКА
ВХОД: граф G, положительное целое число k
СВОЙСТВО: G содержит множество k попарно смежных вершин.

4. УПАКОВКА МНОЖЕСТВ
ВХОД: семейство множеств $\{S_j\}$, положительное целое число l
СВОЙСТВО: $\{S_j\}$ содержит l попарно непересекающихся множеств.

5. ВЕРШИННОЕ ПОКРЫТИЕ
ВХОД: граф G', положительное целое число l
СВОЙСТВО: Существует множество $R \subseteq N'$, такое, что $|R| \leqslant k$ и каждое ребро графа G инцидентно некоторой вершине из R.

6. ПОКРЫТИЕ МНОЖЕСТВАМИ
ВХОД: конечное семейство конечных множеств $\{S_j\}$, положительное целое число k
СВОЙСТВО: Существует подмножество $\{T_h\} \subseteq \{S_j\}$, содержащее $\leqslant k$ множеств, такое, что $\bigcup T_h = \bigcup S_j$.

7. МНОЖЕСТВО ВЕРШИН, РАЗРЕЗАЮЩИХ КОНТУРЫ
ВХОД: ориентированный граф H, положительное целое число k
СВОЙСТВО: существует множество $R \subseteq V$, такое, что каждый контур графа H содержит вершину из R.

8. МНОЖЕСТВО ДУГ, РАЗРЕЗАЮЩИХ КОНТУРЫ
ВХОД: ориентированный граф H, положительное целое число k

СВОЙСТВО: существует множество $S \subseteq E$, такое, что каждый контур графа H содержит дугу из S.

9. ГАМИЛЬТОНОВ КОНТУР
ВХОД: Ориентированный граф H
СВОЙСТВО: H содержит контур, включающий каждую вершину ровно один раз.

10. ГАМИЛЬТОНОВ ЦИКЛ
ВХОД: граф G
СВОЙСТВО: G имеет цикл, включающий каждую вершину ровно один раз.

11. 3-ВЫПОЛНИМОСТЬ (ВЫПОЛНИМОСТЬ НЕ БОЛЕЕ ЧЕМ С 3 БУКВАМИ В КАЖДОЙ СКОБКЕ)
ВХОД: скобки $D_1, D_2, \ldots, D_r$, каждая из которых содержит не более 3 букв из множества букв $\{u_1, u_2, \ldots, u_m\} \cup \cup \{\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_m\}$
СВОЙСТВО: конъюнкция $D_1 \& \ldots \& D_r$ выполнима.

12. ХРОМАТИЧЕСКОЕ ЧИСЛО
ВХОД: граф G, положительное целое число k
СВОЙСТВО: существует функция Φ: $N \to Z_k$, такая, что если u и v смежны, то $\Phi(u) \neq \Phi(v)$.

13. ПОКРЫТИЕ КЛИКАМИ
ВХОД: граф G', положительное целое число l
СВОЙСТВО: N' есть объединение l или меньшего числа клик.

14. ТОЧНОЕ ПОКРЫТИЕ
ВХОД: семейство $\{S_j\}$ подмножеств множества $\{u_i, i = 1, 2, \ldots, t\}$
СВОЙСТВО: существует подсемейство $\{T_h\} \subseteq \{S_j\}$, такое, что множества T_h не пересекаются и $\cup T_h = \cup S_j = \{u_i, i = 1, 2, \ldots, t\}$.

15. МНОЖЕСТВО ПРЕДСТАВИТЕЛЕЙ
ВХОД: семейство $\{U_i\}$ подмножеств множества $\{s_j, j = 1, 2, \ldots, r\}$
СВОЙСТВО: Существует множество W, такое, что для каждого i $|W \cap U_i| = 1$.

16. ДЕРЕВО ШТЕЙНЕРА
ВХОД: граф G, $R \subseteq N$, весовая функция w: $A \to Z$, положительное целое число k
СВОЙСТВО: G имеет поддерево веса $\leqslant k$, множество вершин которого содержит R.

17. 3-МЕРНОЕ ПАРОСОЧЕТАНИЕ
ВХОД: множество $U \subseteq T \times T \times T$, где T — конечное множество
СВОЙСТВО: Существует множество $W \subseteq U$, такое, что $|W| = |T|$ и любые два его элемента отличаются по всем координатам.

18. РЮКЗАК
ВХОД: $(a_1, a_2, \ldots, a_r, b) \in Z^{n+1}$
СВОЙСТВО: $\Sigma a_j x_j = b_j$ имеет 0-1 решение.

19. УПОРЯДОЧЕНИЕ РАБОТ
ВХОД: «вектор времени выполнения» $(T_1, \ldots, T_p) \in Z^p$,
«вектор сроков» $(D_1, \ldots, D_p) \in Z^p$,
«вектор штрафов» $(P_1, \ldots, P_p) \in Z^p$,
положительное целое число k
СВОЙСТВО: Существует перестановка π p-множества $\{1, 2, \ldots p\}$, такая, что $\sum_{j=1}^{p} d_j \leqslant k$, где $d_j = \{p_{\pi(j)}$, если $T_{\pi(j)} + T_{\pi(2)} + \ldots + T_{\pi(j)} > D_{\pi(j)}$.

20. РАЗБИЕНИЕ
ВХОД: $(c_1, c_2, \ldots, c_3) \in Z^s$
СВОЙСТВО: Существует такое множество $I \subseteq \{1, 2, \ldots, s\}$, что $\sum_{h \in I} c_h = \sum_{h \notin I} c_h$.

21. МАКСИМАЛЬНЫЙ РАЗРЕЗ
ВХОД: граф G, весовая функция w: $A \to Z$, положительное целое число W
СВОЙСТВО: Существует множество $S \subseteq N$, такое, что

$$\sum_{\substack{\{u, v\} \in A, u \in S \\ v \notin S}} w(\{u, v\}) \geqslant W.$$

Ясно, что все эти проблемы (или, более точно, их кодировки в Σ^*) принадлежат классу NP. Приступим к осуществлению ряда явных сведений, показывающих, что «выполнимость» сводится к любой из перечисленных проблем. Рисунок 1 (см. стр. 30) показывает структуру множества сведений. Каждая линия рисунка указывает сводимость верхней проблемы к нижней.

Для доказательства сводимости некоторого множества $T \subseteq D$ к множеству $T' \subseteq D'$ мы определим функцию F: $D \to D'$, удовлетворяющую условиям леммы 2. Всякий раз читателю придется затратить некоторые усилия, чтобы убедиться, что F удовлетворяет этим условиям.

ВЫПОЛНИМОСТЬ $\propto$ 0-1 ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ [1])

$$c_{ij}=\begin{cases} 1, & \text{если } x_j \in C_i, \\ -1, & \text{если } \bar{x}_j \in C_i, \\ 0, & \text{в остальных случаях.} \end{cases} \qquad \begin{matrix} i=1,\ 2,\ \ldots,\ p, \\ j=1,\ 2,\ \ldots,\ n, \end{matrix}$$

$b_i = 1 -$ (число переменных с отрицанием в C_i), $i = 1, 2, \ldots, p$.

ВЫПОЛНИМОСТЬ $\propto$ КЛИКА

$$N = \{\langle \sigma, i\rangle \mid \sigma \text{ есть буква, встречающаяся в } C_i\},$$

$$A = \{\{\langle \sigma, i\rangle, \langle \delta, j\rangle\} \mid i \neq j,\ \sigma \neq \bar{\delta}\},$$

$k = p$ число предложений.

КЛИКА $\propto$ УПАКОВКА МНОЖЕСТВ

Положим $N = \{1, 2, \ldots, n\}$. Элементами множеств $S_1, \ldots, S_n$ являются те двухэлементные множества вершин $\{i, j\}$, которые не принадлежат A.

$$S_i = \{\{i, j\} \mid \{i, j\} \notin A\},\ i = 1, 2, \ldots, n,$$

$$l = k.$$

КЛИКА $\propto$ ВЕРШИННОЕ ПОКРЫТИЕ

$$G' \text{ есть дополнение } G,$$

$$l = |N| - k.$$

ВЕРШИННОЕ ПОКРЫТИЕ $\propto$ ПОКРЫТИЕ МНОЖЕСТВ

Положим $N' = \{1, 2, \ldots, n\}$. Элементы — это дуги G', S_j — множество дуг, инцидентных вершине j, $k = l$.

ВЕРШИННОЕ ПОКРЫТИЕ $\propto$ МНОЖЕСТВО ВЕРШИН, РАЗРЕЗАЮЩИХ КОНТУРЫ

$$V = N',$$

$$E = \{\langle u, v\rangle \mid \{u, v\} \in A'\},$$

$$k = l.$$

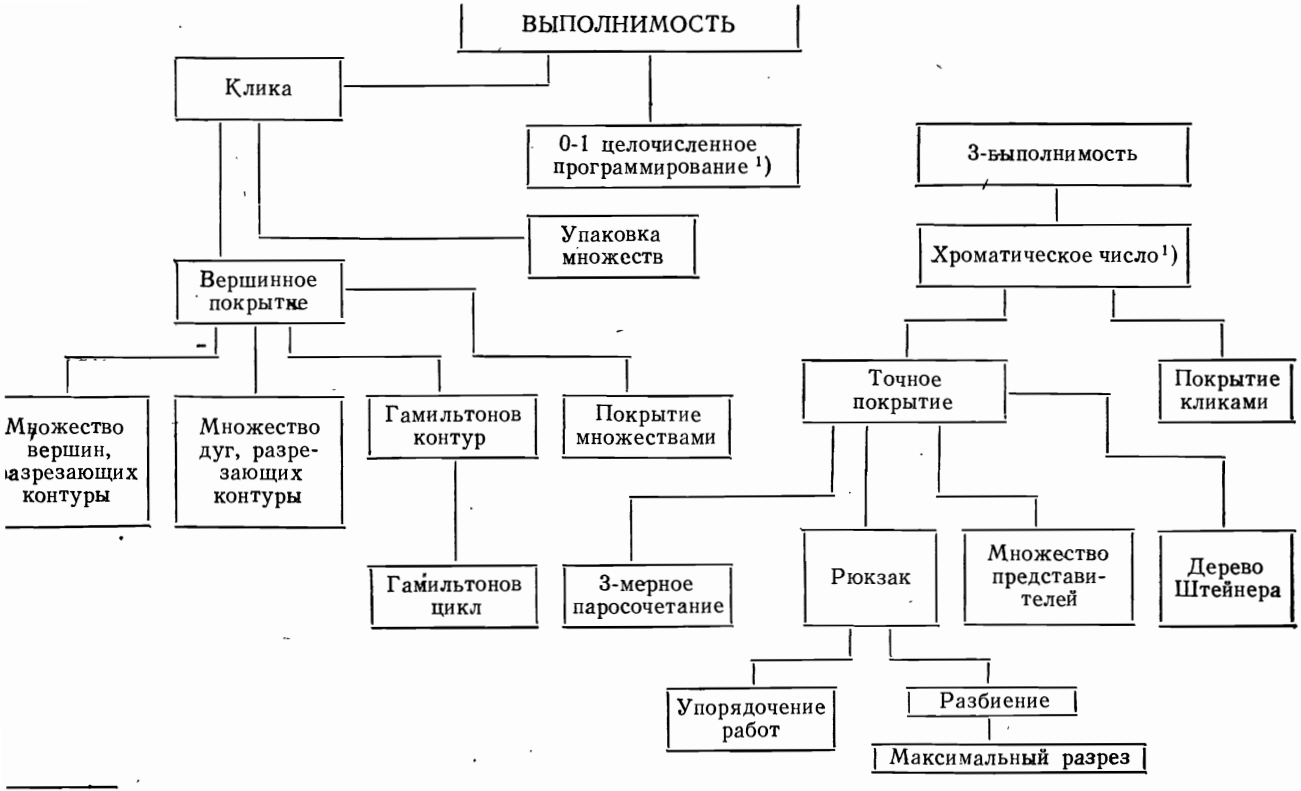
ВЕРШИННОЕ ПОКРЫТИЕ $\propto$ МНОЖЕСТВО ДУГ, РАЗРЕЗАЮЩИХ КОНТУРЫ

$$V = N' \times \{0, 1\},$$

$$F = \{\langle\langle u, 0\rangle, \langle u, 1\rangle\rangle \mid u \in N'\} \cup \{\langle\langle u, 1\rangle, \langle v, 0\rangle \mid \{u, v\} \in A'\},$$

$$k = l.$$

[1]) См. замечания редактора.



[1]) См. замечания редактора.

Рис. 1.

ВЕРШИННОЕ ПОКРЫТИЕ $\propto$ ГАМИЛЬТОНОВ КОНТУР

Без ограничения общности положим $A' = Z_m$.

$V = \{a_1, a_2, \ldots, a_l\} \cup \{\langle u, i, \alpha\rangle \mid u \in N'$ инцидентно $i \in A'$ и $\alpha \in \{0, 1\}\}$,

$F = \{\{\langle u, i, 0\rangle, \langle u, i, 1\rangle\} \mid \langle u, i, 0\rangle \in V\} \cup$

$\cup \{\{\langle u, i, \alpha\rangle, \langle v, i, \alpha\rangle\} \mid u$ и v инцидентны i, $i \in A'$ $\alpha \in \{0, 1\}\}$

$\cup \{\{\langle u, i, 1\rangle, \langle u, j, 0\rangle\} \mid i$ и j инцидентны u и $\neg\exists h$, $i < h < j$, такое, что h инцидентно $u\}$

$\cup \{\{\langle u, i, 1\rangle, a_f\rangle \mid 1 \leqslant f \leqslant l$ и $\neg\exists h > i$ такое, что h инцидентно $u\}$

$\cup \{\{a_f, \langle u, i, 0\rangle\} \mid 1 \leqslant f \leqslant l$ и $\neg\exists h < i$ такое, что h инцидентно $u\}$

ГАМИЛЬТОНОВ КОНТУР $\propto$ ГАМИЛЬТОНОВ ЦИКЛ

$N = V \times \{0, 1, 2\}$,

$A = \{\{\{\langle u, 0\rangle, \langle u, 1\rangle\}, \{\langle u, 1\rangle, \langle u, 2\rangle\}\} \mid u \in V\} \cup$

$\cup \{\{\langle u, 2\rangle, \langle v, 0\rangle\} \mid \langle u, v\rangle \in E\}$.

ВЫПОЛНИМОСТЬ $\propto$ 3-ВЫПОЛНИМОСТЬ

Заменим скобку $\sigma_1 \vee \sigma_2 \vee \ldots \vee \sigma_m$, где σ_i буквы и $m > 3$, на [1])

$$(\sigma_1 \vee \sigma_2 \vee u_1)(\sigma_3 \vee \ldots \vee \sigma_m \vee \bar{u}_1)(\bar{\sigma}_3 \vee u_1) \ldots (\bar{\sigma}_m \vee u_1),$$

где u_1 — новая переменная. Повторим это преобразование до тех пор, пока ни одна из скобок не будет иметь не более чем три буквы.

3-ВЫПОЛНИМОСТЬ $\propto$ ХРОМАТИЧЕСКОЕ ЧИСЛО [2])

Без ограничения общности можно считать, что $m \geqslant 4$.

$N = \{u_1, u_2, \ldots, u_m\} \cup \{\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_m\} \cup \{v_1, v_1, \ldots, v_m\} \cup$

$\cup \{D_1, D_2, \ldots, D_r\}$,

$A = \{\{u_i, \bar{u}_i\} \mid i = 1, 2, \ldots, n\} \cup \{\{v_i, v_j\} \mid i \neq j\} \cup$

$\cup \{\{v_i, x_j\} \mid i \neq j\} \cup \{\{v_i, \bar{x}_j\} \mid i \neq j\} \cup \{\{u_i, D_f\} \mid u_i \notin D_f\} \cup$

$\cup \{\{\bar{u}_i, D_f\} \mid \in D_f\}$,

$k = r + 1$.

ХРОМАТИЧЕСКОЕ ЧИСЛО $\propto$ ПОКРЫТИЕ КЛИКАМИ

G' — дополнение G,

$l = k$.

---

[1]) На самом деле можно заменить $\sigma_1 \vee \sigma_2 \vee \ldots \vee \sigma_m$ формулой $(\sigma_1 \vee \sigma_2 \vee u_1)(\sigma_3 \vee \ldots \vee \sigma_m \vee u_1)$. — *Прим. ред.*

[2]) См. замечания редактора.

ХРОМАТИЧЕСКОЕ ЧИСЛО $\propto$ ТОЧНОЕ ПОКРЫТИЕ

Множеством элементов является множество

$$N \cup A \cup \{\langle u, e, f\rangle \mid e \text{ инцидентно } u \text{ и } 1 \leqslant f \leqslant k\}.$$

Множества S_j определяются следующим образом: для каждого f, $1 \leqslant f \leqslant k$, и любого $u \in N$

$$\{u\} \cup \{\langle u, e, f\rangle \mid u \text{ инцидентно } e\},$$

для каждого $e \in A$ и любой пары f_1, f_2, такой, что $1 \leqslant f_1 \leqslant k$, $1 \leqslant f_2 \leqslant k$ и $f_1 \neq f_2$,

$$\{e\} \cup \{\langle u, e, f\rangle \mid f \neq f_1\} \cup \{\langle v, e, g\rangle \mid g \neq f_2\},$$

где e инцидентно двум вершинам u и v.

ТОЧНОЕ ПОКРЫТИЕ $\propto$ МНОЖЕСТВО ПРЕДСТАВИТЕЛЕЙ

Проблема множества представителей имеет множествами U_i и элементами s_j, такие, что $s_j \in U_i \Leftrightarrow u_i \in S_j$

ТОЧНОЕ ПОКРЫТИЕ $\propto$ ДЕРЕВО ШТЕЙНЕРА

$$N = \{n_0\} \cup \{S_j\} \cup \{u_i\},$$
$$R = \{n_0\} \cup \{u_i\},$$
$$A = \{\{n_0, S_j\}\} \cup \{\{S_j, u_i\} \mid u_i \in S_j\},$$
$$W = (\{n_0, S_j\}) = |S_j|,$$
$$W = (\{S_j, u_i\}) = 0,$$
$$k = |\{u_i\}|.$$

ТОЧНОЕ ПОКРЫТИЕ $\propto$ 3-МЕРНОЕ ПАРОСОЧЕТАНИЕ

Без ограничения общности положим $|S_j| \geqslant 2$ для любого j. Пусть $T = \{\langle i, j\rangle \mid u_i \in S_j\}$. Пусть α — произвольная взаимно однозначная функция из $\{n_i\}$ в T. Пусть π: $T \to T$ — перестановка, такая, что для любого фиксированного j, $\{\langle i, j\rangle \mid u_i \in S_j\}$ есть цикл π.

$$U = \{\langle \alpha(u_i), \langle i, j\rangle, \langle i, j\rangle\rangle \mid \langle i, j\rangle \in T\} \cup \{\langle \beta, \sigma, \pi(\sigma)\rangle \mid \text{для всех } i,\ \beta \neq \alpha(u_i)\}.$$

ТОЧНОЕ ПОКРЫТИЕ $\propto$ РЮКЗАК

Пусть $d = |\{S_j\}| + 1$. Пусть $e_{ji} = \begin{cases} 1, & \text{если } u_i \in S_j, \\ 0, & \text{если } u_i \notin S_j. \end{cases}$

Пусть $r = |\{S_j\}|$, $a_j = \sum e_{ji} d^{i-1}$ и $b = \dfrac{d^t - 1}{d - 1}$.

РЮКЗАК $\propto$ УПОРЯДОЧЕНИЕ РАБОТ

$$p = r, \quad T_i = P_i = a_i, \quad D_i = b.$$

РЮКЗАК $\propto$ РАЗБИЕНИЕ

$$s = r + 2,$$
$$c_i = a_i, \quad i = 1, 2, \ldots, r,$$
$$c_{r+1} = b_{r+1},$$
$$c_{r+2} = \left(\sum_{i=1}^{r} a_i\right) + 1 - b.$$

РАЗБИЕНИЕ $\propto$ МАКСИМАЛЬНЫЙ РАЗРЕЗ

$$N = \{1, 2, \ldots, s\},$$
$$A = \{\{i, j\} \mid i \in N, \ j \in N, \ i \neq j\},$$
$$W = (\{i, j\}) = c_i c_j,$$
$$W = \left[\tfrac{1}{4} \sum c_i^2\right].$$

Некоторые из описанных здесь сведéний не принадлежат автору этих строк. Кук [1] показал, что ВЫПОЛНИМОСТЬ $\propto$ 3-ВЫПОЛНИМОСТЬ. Сведение ВЫПОЛНИМОСТЬ $\propto$ КЛИКА намечено у Кука [1] и было известно также Рейтеру. Сведение ВЕРШИННОЕ ПОКРЫТИЕ $\propto$ МНОЖЕСТВО ВЕРШИН, РАЗРЕЗАЮЩИХ КОНТУРЫ было найдено на Семинаре по теории алгоритмов Отделения вычислительных наук Корнельского университета. Сведение ВЕРШИННОЕ ПОКРЫТИЕ $\propto$ $\propto$ МНОЖЕСТВО ДУГ, РАЗРЕЗАЮЩИХ КОНТУРЫ было найдено Лоулером и автором, и Лоулер нашел сведение ТОЧНОЕ ПОКРЫТИЕ $\propto$ 3-РАЗМЕРНОЕ ПАРОСОЧЕТАНИЕ. Автор обнаружил, что проблема точного покрытия сводится к проблеме коммивояжера на ориентированном графе, дуги которого имеют длину 0 или 1. Усовершенствовав технику, использованную в этой конструкции, Тарьян показал, что ТОЧНОЕ ПОКРЫТИЕ $\propto$ ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ЦИКЛ и, независимо, Лоулер показал, что ВЕРШИННОЕ ПОКРЫТИЕ $\propto$ ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ЦИКЛ. Сведение ОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ЦИКЛ $\propto$ НЕОРИЕНТИРОВАННЫЙ ГАМИЛЬТОНОВ ЦИКЛ нашел Тарьян.

Ниже мы приводим три проблемы из теории автоматов и теории языков, к которым сводятся все полные проблемы. Неизвестно, являются ли эти проблемы полными, их принадлежность к *NP* в настоящее время кажется сомнительной. Читатель, незнакомый с теорией автоматов и языков, может найти необходимые определения у Хопкрофта и Ульмана [3].

ЭКВИВАЛЕНТНОСТЬ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ

ВХОД: пара регулярных выражений над алфавитом {0, 1}

СВОЙСТВО: эти два выражения определяют один и тот же язык

ЭКВИВАЛЕНТНОСТЬ НЕДЕТЕРМИНИРОВАННЫХ КОНЕЧНЫХ АВТОМАТОВ

ВХОД: пара недетерминированных конечных автоматов с входом $\{0, 1\}$

СВОЙСТВО: эти два автомата определяют один и тот же язык.

КОНТЕКСТНО-ЗАВИСИМОЕ РАСПОЗНАВАНИЕ

ВХОД: Контекстно-зависимая грамматика Γ и цепочка x.

СВОЙСТВО: x принадлежит языку, порожденному Γ. Первым делом покажем, что

3-ВЫПОЛНИМОСТЬ $\propto$ ЭКВИВАЛЕНТНОСТЬ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ

Сведение проведем в два этапа. Сначала построим пару регулярных выражений в алфавите

$$\Delta = \{u_1, u_2, \ldots, u_n; \bar{u}_1, \bar{u}_2, \ldots, \bar{u}_n\},$$

затем преобразуем эту пару в одно регулярное выражение над алфавитом $\{0, 1\}$.

Первое регулярное выражение есть $\Delta^n\Delta^*$ (более точно, Δ выписывается как $(u_1 + u_2 + \ldots + u_n + \bar{u}_1 + \ldots + \bar{u}_n)$ и Δ^n представляет n экземпляров выражения Δ, связанных конкатенацией). Второе регулярное выражение есть

$$\Delta^n\Delta^* \cup \bigcup_{i=1}^{n} (\Delta^* u_i \Delta^* \bar{u}_i \Delta^* \cup \Delta^* \bar{u}_i \Delta^* u_i \Delta^*) \cup \bigcup_{h=1}^{r} \theta(D_h),$$

где

$$\theta(D_h) = \begin{cases} \Delta^*\bar{\sigma}_1\Delta^*, & \text{если } D_h = \sigma_1, \\ \Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_2\Delta^* \cup \Delta^*\bar{\sigma}_2\Delta^*\bar{\sigma}_1\Delta^*, & \text{если } D_h = \sigma_1 \cup \sigma_2, \\ \Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_2\Delta^* \cup \Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_3\Delta^*\bar{\sigma}_2\Delta^* \cup \Delta^*\bar{\sigma}_2\Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_3\Delta^* \cup \\ \quad \cup \Delta^*\bar{\sigma}_2\Delta^*\bar{\sigma}_3\Delta^*\bar{\sigma}_1\Delta^* \cup \Delta^*\bar{\sigma}_3\Delta^*\bar{\sigma}_1\Delta^*\bar{\sigma}_2\Delta^* \cup \Delta^*\bar{\sigma}_3\Delta^*\bar{\sigma}_2\Delta^*\bar{\sigma}_1\Delta^*, \\ \qquad \text{если } D_h = \sigma_1 \cup \sigma_2 \cup \sigma_3. \end{cases}$$

Пусть теперь m — наименьшее положительное число $\geqslant \log_2|\Delta|$, и пусть Φ — взаимно однозначное отображение Δ в $\{0, 1\}^m$. Заменяем каждое регулярное выражение регулярным выражением над $\{0, 1\}$, производя замену $a \to \Phi(a)$ для каждого вхождения каждого элемента Δ.

ЭКВИВАЛЕНТНОСТЬ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ $\propto$ ЭКВИВАЛЕНТНОСТЬ НЕДЕТЕРМИНИРОВАННЫХ КОНЕЧНЫХ АВТОМАТОВ

Существуют стандартные алгоритмы (Саломаа [5]), превращающие регулярное выражение в эквивалентный недетер-

минированный автомат за полиномиально ограниченное число шагов.

В заключение мы покажем, что для любого $L \in NP$
$L \propto$ КОНТЕКСТНО-ЗАВИСИМОЕ РАСПОЗНАВАНИЕ.

Предположим, L распознаваем за время $p(\cdot)$ на детерминированной машине Тьюринга. Тогда язык $\tilde{L}$ в алфавите $\{0, 1, \#\}$ принимается недетерминированным линейно-ограниченным автоматом, который моделирует машину Тьюринга:

$$Z = \{\#^{p(\lg(x))} x \#^{p(\lg(x))} \mid x \in L\}.$$

Следовательно, $\tilde{L}$ является контекстно-зависимым языком и имеет грамматику Γ. Таким образом, $x \in L$ тогда и только тогда, когда

$$\tilde{\Gamma},\ \#^{p(\lg(x))} x \#^{p(\lg(x))}$$

— допустимый вход для КОНТЕКСТНО-ЗАВИСИМОГО РАСПОЗНАВАНИЯ.

В заключение дадим список следующих важных проблем из NP, для которых неизвестно, полны они или нет.

ИЗОМОРФИЗМ ГРАФОВ
ВХОД: графы G и G'
СВОЙСТВО: GG' изоморфны.

НЕПРОСТОТА
ВХОД: положительное целое число k
СВОЙСТВО: k составное (не простое) число.

ЛИНЕЙНЫЕ НЕРАВЕНСТВА
ВХОД: целочисленная матрица C, целочисленный вектор d
СВОЙСТВО: $Cx \geqslant d$ имеет рациональное решение.

ПРИЛОЖЕНИЕ 1

Система обозначений и терминология, использованные в описании проблем.

Пропозициональное исчисление

$x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_m$ пропозициональные переменные

$\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n, \bar{u}_1, \bar{u}_2, \ldots, \bar{u}_m$ отрицания пропозициональных переменных

σ, σ_i буквы

$C_1, C_2, \ldots, C_p D_1, D_2, \ldots, D_r$ скобки (элементарные дизъюнкции)

$$C_k \subseteq \{x_1, x_2, \ldots, x_n, \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\},$$

$$D_2 \subseteq \{u_1, u_2, \ldots, u_m, \bar{u}_1, \bar{u}_2, \ldots, \bar{u}_m\}.$$

Никакая скобка не содержит переменной вместе с ее отрицанием.

Скаляры, векторы, матрицы

Z	положительные целые числа
Z^p	множество p-ок положительных целых чисел
Z_p	множество $\{0, 1, \ldots, p-1\}$
k, W	элементы Z
$\langle x, y\rangle$	упорядоченная пара $\langle x, y\rangle$
$(a_i)\,(y_i)\,d$	векторы с неотрицательными целочисленными координатами
$(c_{ij})\,C$	матрицы с целочисленными элементами.

Графы и ориентированные графы

$G=(N, A)\ G'=(N', A')$	конечные графы
N, N'	множества вершин
A, A'	множества дуг
s, t, u, v	вершины
$e, \{u, v\}$	дуги
$(X\bar{X})=\{\{u, v\} \mid u \in X, v \in \bar{X}\}$	разрез

Если $s \in X$ и $t \in \bar{X}$, то $(X, \bar{X})$ есть $s-t$ разрез. w: $A \to Z$, w': $A^* \to Z$ весовые функции

Вес подграфа есть сумма весов его дуг

$H=(V, E)$	ориентированный граф
V	множество вершин
E	множество дуг
$e, \langle u, v\rangle$	дуги

Множества

$\varnothing$	пустое множество
$\lvert S\rvert$	число элементов конечного множества S
$\{S_l\}\{T_h\}\{u_i\}$	конечные семейства конечных множеств.

ЗАМЕЧАНИЯ РЕДАКТОРА ПЕРЕВОДА

1. Сведение ВЫПОЛНИМОСТЬ $\propto$ 0-1 ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ неверно. Например, формула $x_1(x_1 \vee x_2)x_2$ выполнима, но соответствующая ей система $x_1 = 1$, $x_1 + x_2 = 1$, $x_2 = 1$ решения не имеет.

Покажем, что РЮКЗАК $\propto$ 0-1 ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ.

РЮКЗАК

ВХОД: $(a_1, a_2, \ldots, a_r, b) \in Z^{n+1}$

СВОЙСТВО: $\Sigma a_j x_j = b_j$ имеет 0-1 решение.

0-1 ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ

ВХОД: целочисленная матрица C и целочисленный вектор d

СВОЙСТВО: Существует вектор x из нулей и единиц, для которого $Cx = d$

Возьмем в качестве матрицы C вектор $(a_1, a_2, \ldots, a_2)$, в качество вектора d число b.

2. Сведение 3-ВЫПОЛНИМОСТЬ $\propto$ ХРОМАТИЧЕСКОЕ ЧИСЛО неверно. Например, формула $(u_1 \vee u_2 \vee u_3)(\bar{u}_1 \vee \bar{u}_2 \vee u_4)$ выполнима, но соответствующий ей граф (N, A) содержит полный подграф с вершинами v_1, v_2, v_3, v_4, не раскрашиваемый в три цвета.

Покажем, что

ПОКРЫТИЕ МНОЖЕСТВАМИ $\propto$ ХРОМАТИЧЕСКОЕ ЧИСЛО.

ПОКРЫТИЕ МНОЖЕСТВАМИ

ВХОД: Конечное семейство $\{S_j\}$ конечных множеств, положительное целое число l.

СВОЙСТВО: Существует подсемейство $\{T_h\} \subseteq S_j$, содержащее $\leqslant l$ множеств, такое, что $UT_h = US_j$.

ХРОМАТИЧЕСКОЕ ЧИСЛО

ВХОД: граф G, положительное целое число l.

СВОЙСТВО: существует функция Φ: $N \to Zl$, такая, что если u и v смежны, то $\Phi(u) \neq \Phi(v)$.

Построим граф $G' = (N, A)$, в котором $N = \cup S_j$, $A = \{(r, t) \mid$ существует S_j, такое, что $r, t \in S_j\}$. Таким образом, множествам соответствуют полные подграфы графа G'. Пусть G'' есть дополнение G''. Пусть G получается из G'' добавлением l изолированных вершин. Тогда подсемейство $\{T_h\} \subseteq \{S_j\}$, содержащее не более l множеств и удовлетворяющее условию $\cup T_h \neq \cup S_j$, существует в том и только том случае, когда вершины графа G можно раскрасить в l цветов так, что смежные вершины раскрашены разными цветами.

ЛИТЕРАТУРА

1. Cook S. A., The complexity of theorem-proving procedures, Proceedings Third Ann. ACM Symposium on the Theory of Computing, Shaker Heights, N.-Y., 1971, 151—158. (Кук С. А. Сложность процедур доказательства теорем, см. наст. сб., стр.5—15.)
2. Edmonds J., Path tress and flowers, *Canad. I. Math.*, VIII, 449—467.
3. Hopcroft J., and Ullman J. D. Formal Languages and Their Relation to Automata, Addison — Wesley, Reading, Massachussets.
4. Kruskal J., On the shortest spanning subtree of a graph and the traveling salesman problem, *Proceedings Amer. Math. Soc.*, **7** (1956), 48—50.
5. Dijkstra E., A note on two problems in connection with graphs, *Num. Math.*, **1** (1959), 269—271.
6. Edmonds J. and Karp R., Theoretical improvements in algorithmic efficiency for network flow problems, *J. A. C. M.*, **19**, № 2 (1972), 248—264.
7. Hall M., Distinct representatives of subsets, *Bull. Amer. Amer. Math. Soc.*, **54** (1948), 922—926.
8. Rogers H., Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York. Русский перевод. Роджерс Х. Теория рекурсивных функций и эффективная вычислимость, М., Мир, 1972.
9. Salomaa A., Theory of Automata, Pergamon Press, Elmsford, New York, 1969.

Изоморфизм планарных графов[1])

Дж. Е. Хопкрофт, П. Е. Тарьян

В статье предлагается алгоритм установления изоморфизма двух планарных графов, который требует $O(V \cdot \log V)$ действий, где V — число вершин в каждом из графов.

1. ВВЕДЕНИЕ

Установление изоморфизма планарных графов является важным частным случаем общей проблемы установления изоморфизма графов. Эта задача возникает при перечислении различных классов планарных графов, а также при решении ряда технических задач. Целесообразно рассмотреть случай планарных графов отдельно, поскольку в настоящее время изучение более общих случаев имеет мало перспектив. Хотя для общей задачи установления изоморфизма существуют хорошие эвристические способы решения, все известные алгоритмы в худших случаях характеризуются числом действий, зависящим асимптотически экспоненциально от числа вершин.

В данной работе описывается эффективный алгоритм установления изоморфизма двух планарных графов. Асимптотически число действий этого алгоритма определяется как $O(V \cdot \log V)$, где V — число вершин в каждом из графов. Здесь как промежуточный результат приводится также алгоритм установления изоморфизма деревьев, характеризующийся линейным числом действий. Некоторые авторы (Эдмондс, Скойнс, Вейнбург, и др.) описывают подобные алгоритмы, но ни один из них не приводит нетривиальных деталей выполнения сортировок, используемых в предлагаемых алгоритмах.

Первые результаты об установлении изоморфизма планарных графов принадлежат Вейнбергу, который разработал эффективный алгоритм определения изоморфизма 3-связных планарных графов, требующий число действий, пропорциональное V^2, а также алгоритмы установления изоморфизма «последовательно-параллельных» графов и деревьев. Эти алгоритмы можно легко скомбинировать для того, чтобы получить полиномиально-ограниченный алгоритм решения задачи в случае произвольной

[1]) Hopcroft J., Tarjan R., Isomorphism of planar graphs, Proc 4d Annual Symp. on the Theory of Computing, Shaker Heigts, 1972, 131—152.

пары планарных графов. Далее, усовершенствованный алгоритм установления изоморфизма 3-связных планарных графов дан Хопкрофтом [1971 А], а затем Хопкрофт и Тарьян [1971 А] описали алгоритм установления изоморфизма планарных графов с оценкой числа действий V^2.

Предлагаемая статья состоит из пяти разделов. Введение и определение необходимых понятий теории графов составляют содержание первого раздела. Во втором разделе описывается алгоритм нахождения разбиения графа на однозначно определенные 3-связные компоненты за число действий, пропорциональное числу ребер графа. В третьем разделе дается алгоритм установления изоморфизма деревьев. В четвертом разделе описывается алгоритм установления изоморфизма 3-связных планарных графов за число действий, пропорциональное $V \cdot \log V$. В пятом описанные алгоритмы объединяются и описывается алгоритм установления изоморфизма произвольной пары планарных графов. В самых плохих случаях число действий алгоритма растет пропорционально $V \cdot \log V$.

Приведем теперь необходимые понятия и сделаем некоторые замечания. Предполагается, что читатель знаком с общепринятыми понятиями теории графов (см. Харари [1969 А]). Граф G определяется конечным множеством вершин V и конечным множеством ребер $\mathscr{E}$. Если ребрами являются неупорядоченные пары вершин, то имеем понятие *неориентированного* графа. Если ребра — упорядоченные пары вершин, то граф называется *ориентированным*. Если (v, w) — ориентированное ребро (дуга), то вершина w называется *началом* дуги, а вершина v — концом. Маршрут (обозначается $v \overset{*}{\Rightarrow} w$) — это последовательность вершин и ребер, идущих из v к w. Маршрут называется *простым*, если все его вершины различные. *Циклом* называется замкнутый маршрут, у которого все ребра различные и только одна вершина повторяется дважды.

Дерево — это связный граф, не содержащий циклов. *Корневым деревом* называется ориентированный граф, обладающий следующими тремя свойствами: 1) существует точно одна вершина, называемая *корнем*, в которую не входит ни одна дуга; 2) для каждой вершины дерева существует последовательность дуг из корня в эту вершину; 3) в любую вершину, за исключением корня, входит точно одна дуга. Будем обозначать дугу (v, w) дерева через $v \to w$. Маршрут (ориентированный), идущий из вершины v в вершину w, обозначим через $v \overset{*}{\to} w$.

Если $v \to w$, то будем называть вершину v *отцом* вершины w, а вершину w — *сыном* вершины v. Если же $v \overset{*}{\to} w$, то вершину v назовем *предком* вершины w, а вершину w — *потомком* вершины v. Каждая вершина по отношению к себе является и

предком, и потомком. Для вершины v дерева T обозначим через T_v поддерево дерева T, все вершины которого являются потомками вершины v в T.

Пусть G — ориентированный граф. Дерево T называется *остовным деревом* графа G, если T — подграф графа G и если T содержит все вершины графа G.

Граф G *2-связен*, если для каждой тройки различных вершин v, w, a существует такой маршрут p: $v \overset{*}{\Rightarrow} w$, что вершина a не принадлежит p. Если существуют такие три различные вершины v, w, a, что вершина a принадлежит любому маршруту p: $v \overset{*}{\Rightarrow} w$, то вершина a называется точкой сочленения графа G. Пусть множество ребер $\mathscr{E}$ графа G разбито на непересекающиеся подмножества $\{\mathscr{E}_i\}$ таким образом, что два ребра находятся в одном подмножестве в том и только том случае, когда эти ребра принадлежат некоторому (одному) циклу. Введем подграфы $G_i = (V_i, \mathscr{E}_i)$, где $\mathscr{E}_i$ — множества ребер i-го подмножества, а V_i — множество всех вершин, инцидентных ребрам из $\mathscr{E}_i$. Тогда: 1) каждый граф G_i 2-связен; 2) ни один из графов G_i не является собственным подграфом какого-нибудь 2-связного подграфа графа G; 3) любая вершина графа G, не являющаяся точкой сочленения, принадлежит точно одному множеству V_i, а каждая точка сочленения принадлежит по крайней мере двум различным множествам; 4) для каждой пары i, j ($i \neq j$) множество $V_i \cap V_j$ содержит не более одной вершины, причем если $|V_i \cap V_j| = 1$, то соответствующая вершина является точкой сочленения. Подграфы G_i графа G называются его *2-связными компонентами*.

Граф G 3-связен, если для любых четырех различных вершин v, w, a, b из V существует такой маршрут p: $v \overset{*}{\Rightarrow} w$, что вершины a, b не принадлежат p. Если найдется такая четверка различных вершин v, w, a, b, что каждый маршрут p: $v \overset{*}{\Rightarrow} w$ содержит или вершину a, или вершину b, то пара a, b называется *2-сочленением* графа G. *n-угольник* — это связный граф, состоящий из цикла с n ребрами. *n-связкой* назовем пару вершин, соединенных n ребрами, строго говоря n-связка не является графом [см. Харари (1969А)]. Это понятие вводится потому, что при нахождении 2-сочленений и разбиений графа на 3-связные компоненты производится удаление компонент и замена их ребрами, в результате чего возникает необходимость рассматривать кратные ребра.

При соответствующей модификации введенного выше определения, например, такой, как в (Татт [1966А]), или при разбиении только 2-связных компонент можно обеспечить единственность выделения 3-связных компонент.

Для получения оценок числа действий алгоритмов используется модель с произвольным порядком выбора. Чтобы не рассматривать специальные детали этой модели, введем следующее обозначение. Если для вектора $\vec{n}$ и функций $t(\vec{n})$, $f(\vec{n})$ существуют такие константы k_1, k_2, что $|t(\vec{n})| \leqslant k_1 \cdot |f(\vec{n})| + k_2$, то будем писать „$t(\vec{n})$ есть $O(f(\vec{n}))$".

Будем использовать несколько вспомогательных алгоритмов. С помощью одного из них, называемого основной сортировкой (radix sort), размещают n целых чисел $x_1, x_2, \ldots, x_n$, каждое из которых заключено между 1 и n, по ящикам, за число действий $O(n)$. Каждое число x_i помещается в ящик x_i, а затем содержимое ящиков извлекается в порядке возрастания, начиная с ящика 1.

В вычислительной машине граф представляется с помощью задания *структуры смежности*, которая состоит из множества списков вершин, смежных с каждой вершиной (для краткости будем говорить «списки смежности»). Список смежности вершины v содержит все вершины w, такие, что ребро (v, w) принадлежит множеству ребер $\mathscr{E}$. Если G — неориентированный граф, то каждое ребро представлено в структуре смежности дважды; если G — ориентированный граф, то каждая его дуга встречается один раз. Для данного графа структура смежности определяется не единственным образом, и структур столько, сколько существует упорядочений ребер, инцидентных каждой вершине.

Представление графа с помощью структуры смежности позволяет ускорить обследование графа. Мы будем пользоваться специальным типом обследования, который называется *просмотром глубины 1*. В просмотре глубины 1 граф предварительно исследуется таким образом, что всегда выбирается ребро, выходящее из вершины, у которой имеются еще не пройденные инцидентные ребра и которую мы достигли позже всех других вершин с таким свойством.

Пусть G — неориентированный граф. При просмотре этого графа каждому ребру приписывается ориентация, в котором оно проходится, когда осуществляется обход графа G. Таким образом граф G становится ориентированным графом G'. Множество ребер, которые входят в новую рассматриваемую вершину, когда она проходится во время просмотра, определяет остовное дерево графа G'. Вообще говоря, дуги графа G', не включенные в остовное дерево, соединяют маршруты этого дерева. Однако если используется просмотр глубины 1, то любая дуга (v, w), не принадлежащая остовному дереву, соединяет вершину v с одним из ее предков w. В этом случае будем называть граф G' *пальмовым деревом*, для краткости *п-деревом*, а дуги графа G',

не принадлежащие остовному дереву, назовем *листьями* этого графа. Дугу (v, w), являющуюся листом графа G', будем обозначать $v \rightarrow\!\!\!\!- w$. Используя структуру смежности для получения ребра, которое инцидентно данной вершине, просмотр глубины I можно реализовать за $O(V, E)$ действий.

2. ОПРЕДЕЛЕНИЕ 3-СВЯЗНОСТИ

Пусть $G = (\mathscr{V}, \mathscr{E})$ — граф, имеющий $|\mathscr{V}| = V$ вершин и $|\mathscr{E}| = E$ ребер. Опишем в этом разделе алгоритм определения 3-связности графа G, который требует $O(V, E)$ действий. Ранее известные алгоритмы требуют большего количества действий, например алгоритм, предложенный в работе (Ариёси, Сиракано, Хироси [1971]), характеризуется $O(V^4)$ действиями. Если использовать определение 3-связной компоненты, которое дано Таттом (см. Татт [1966]), или другое аналогичное определение, то предлагаемый здесь алгоритм позволяет разбить граф на 3-связные компоненты за $O(V, E)$ действий. (Определение Татта обладает тем преимуществом, что разбиение графа на 3-связные компоненты единственно; единственность разбиения существенным образом используется для решения задачи об изоморфизме планарных графов.)

Можно предположить, что $V > 4$ и что граф G не имеет вершин степени 2, поскольку если $V \leqslant 4$ или в G имеется вершина степени 2, то вопрос о трехсвязности решается немедленно. Далее можно предположить, что G — 2-связный граф. В статье (Хопкрофт, Тарьян [1971 А]) описан метод разбиения графа на 2-связные компоненты за $O(V, E)$ действий.

Алгоритм определения 3-связности состоит из трех просмотров глубины 1. С помощью первого просмотра для графа G строится n-дерево P и находится необходимая информация о листьях этого дерева. Полученная информация используется для построения структуры смежности A n-дерева P. Во втором просмотре структура A используется для выбора исследуемых ребер, а также находится последующая необходимая информация о n-дереве P. В третьем просмотре определяется, существуют ли в графе G 2-сочленения.

Предположим, что граф G обследован с помощью процесса типа просмотра глубины 1 и что вершины графа G занумерованы в этом порядке, в котором они проходятся во время просмотра. Отождествим вершины с приписанными им числами. Пусть P — n-дерево, построенное с помощью выполненного просмотра. Введем для вершины $v \in \mathscr{V}$ две характеристики $L_1(v) = \min(\{v\} \cup \{w \mid v^* \Rightarrow w\})$ и $L_2(v) = \min[\{v\} \cup (\{w \mid v^* \rightarrow\!\!\!\!- \cdots \rightarrow\!\!\!\!- w\} - L_1(v))]$. Таким образом, $L_1(v)$ — это наименьшая вершина, в которую можно попасть из вершины v, пройдя по $l \geqslant 0$ дугам n-дерева P, завершающимся самое большее одним листом. $L_2(v)$ — номер

шины, следующий после наименьшей вершины, достигаемой таким же точно образом из v. Если не выполнено $L_1(v) = L_2(v) = v$, то $L_1(v) < L_2(v)$. Номера вершин и значения $L_1(v)$ и $L_2(v)$ для всех вершин можно легко определить во время первого просмотра графа G. Для дуги $v \to w$ из P положим $\varnothing((v, w)) = L_1(w)$. Если $v \not\to w$, то положим $\varnothing((v, w)) = w$. Пусть A — структура смежности n-дерева P, такая, что списки смежности в A упорядочены по значениям функции $\varnothing$. (Каждый элемент в списке смежности соответствует некоторой дуге n-дерева P; все элементы списка нужно упорядочить по значениям функции $\varnothing$ на соответствующих дугах.) Такую структуру смежности A можно построить, используя только одну основную сортировку дуг n-дерева P, см. (Тарьян [1972 B]). Более того, структура A зависит только от порядка значений $L_1(v)$ и не зависит от способа нумерации. То есть если вершины n-дерева P занумерованы числами от 1 до V произвольным образом, но так, что из $v \to w$ для номеров вершин выполняется $N(v) < N(w)$, и если значения функции $L_1(v)$ вычислены по новым номерам, то возможные структуры смежности, которые удовлетворяют новому упорядочению значений $L_1(v)$, совпадают со структурами, которые удовлетворяют старому упорядочению. Это утверждение легко доказать, см. (Тарьян [1972 B]).

Во втором просмотре дуги обследуются в порядке, который определяется структурой смежности A, начиная с вершины, которая была начальной и в первом просмотре. Вершины нумеруются числами от V до 1, каждая из них получает номер при последнем прохождении во время просмотра. Из процедуры приписывания номеров следует, что если $v \to w$, то $N(v) < N(w)$, и если $v \to w_1$, $v \to w_2$, то $N(w_1) > N(w_2)$ при условии, что дуга (v, w_1) проходится раньше дуги (v, w_2) во время второго просмотра. Теперь уже будем отождествлять вершины с номерами, приписанными вершинам во время второго просмотра. По новой нумерации заново вычисляются значения $L_1(v)$ и $L_2(v)$. Введем теперь два множества чисел, важных для последующего изложения. Если $u \to v$ принадлежит n-дереву P, то положим $h(v) = \max(\{u\} \cup \{w \mid v \xrightarrow{*} w \in w \not\to u\})$. Значение $h(v)$ является наибольшей (по номеру) концевой вершиной листа, который начинается потомком, а кончается отцом вершины v. Обозначим через $H(v)$ наибольшего потомка вершины v. Величины $h(v)$ и $H(v)$ для каждой вершины v вычисляются во время второго просмотра.

После завершения второго просмотра получаем n-дерево P графа G, упорядоченное в соответствии со структурой смежности A. Мы также определили несколько множеств чисел, которые приписаны вершинам n-дерева P. Теперь по этим данным можно определить все 2-сочленения графа G.

Лемма 1. *Пусть P — n-дерево 2-связного графа G, полученное просмотром глубины 1, a $\{a, b\}$ — 2-сочленение графа G, такое, что $a < b$. Тогда в остовном дереве T n-дерева P существует путь $a \xrightarrow{*} b$.*

Доказательство. Предположим, что вершина b не является потомком вершины a в n-дереве P. Пусть $D(v)$ — множество потомков вершины v, принадлежащей остовному дереву T. Подграф графа G, порожденный множеством вершин $W = \mathscr{V} - (D(a) \cap D(b))$, связен. Если вершина v является сыном вершины a или вершины b, то вершины из $D(v)$ смежны со всеми вершинами из $D(v) \cup W \cup \{a, b\}$. Но так как $\{a, b\}$ — 2-сочленение, то или вершина a, или вершина b должна быть точкой сочленения, что невозможно, поскольку граф G 2-связен.

Утверждение леммы 1 дает необходимое и достаточное условие того, чтобы пара вершин $\{a, b\}$ являлась 2-сочленением. Если $v \to w$ и вершина w есть первый элемент в списке смежности вершины v, то назовем вершину w первым сыном вершины v. Если в n-дереве P существует путь $v \xrightarrow{*} w$ и каждая вершина этого пути, отличная от v, является *первым* сыном своего отца, то вершина w называется первым потомком вершины v. Любая вершина является первым потомком по отношению к себе.

Лемма 2. *Пусть P — n-дерево 2-связного графа G, полученное просмотром глубины 1. Пусть также (a, b) — 2-сочленение и $a < b$. Тогда справедливо одно из следующих двух предложений:*

(1) *Существуют такие различные вершины $r \neq a, b$ и $s \neq a, b$, что $b \to r$, $L_1(r) = a$, $L_2(r) \geqslant b$ и вершина s не является потомком вершины r (пару вершин (a, b) будем называть 2-сочленением 1-го типа).*

(2) *Существует вершина $r \neq b$, такая, что: $a \to r \xrightarrow{*} b$; вершина b — первый потомок вершины r; $a \neq 1$; для каждого листа $i \to j$, у которого $r \leqslant i < j$, выполняется $a \leqslant j$, а для любого листа $i \to j$, у которого $a < j < b$ и $b \to w \xrightarrow{*} i$, выполняется $L_1(w) \geqslant a$ (такая пара вершин (a, b) называется 2-сочленением 2-го типа).*

Обратно, любая пара вершин (a, b), удовлетворяющая (1) или (2), является 2-сочленением графа G.

Доказательство. Обратное утверждение легко доказывается, поэтому будем доказывать только прямое утверждение. Для этого предположим, что пара вершин (a, b) является 2-сочленением графа G и что $a < b$. Из леммы 1 имеем, что в остовном дереве T n-дерева P существует путь $a \xrightarrow{*} n$. Пусть $b_1, b_2, \ldots, b_n$ — сыновья вершины b, выписанные в том порядке, в котором они встречаются в списке смежности A_b вершины b.

Пусть также $a \to v \xrightarrow{*} b$. Для вершины w n-дерева P обозначим через $D(w)$ множество потомков вершины w, принадлежащих остовному дереву T. Положим $X = D(v) - D(b)$ и $W = V - D(a)$. Если вершина $w \neq v$ является сыном вершины a, то некоторая вершина из $D(w)$ смежна с какой-либо вершиной из W, так как граф G 2-связен и вершины, принадлежащие множеству $D(w)$, смежны только с вершинами из $D(w) \cup \{a\} \cup W$. Вершины из множества $D(b_i)$ смежны только с вершинами, принадлежащими множеству $D(b_i) \cup \{a, b\} \cup X \cup W$.

Если удаление вершин a, b приводит к отделению некоторого подграфа $D(b_i)$ от остальной части графа G, то легко показать, что пара (a, b) удовлетворяет предположению (1), когда $r = b_i$, а s — некоторая вершина из оставшейся части графа G. Если это не так, то удаление вершин a, b должно отделять X и, возможно, некоторую часть подграфа $D(b_i)$ от W и остальной части подграфа $D(b_i)$. К тому же поскольку $L_1(b_1) \leqslant L_1(b_2) \leqslant \ldots \ldots \leqslant L_1(b_n)$, то существует такое $k_0 \geqslant 1$, что подграфы $W, D(b_1), \ldots, D(b_{k_0})$ отделены от подграфов $X, D(b_{k+1}), \ldots, D(b_n)$. На самом деле, это число равно $k_0 = \max\{i \mid L_1(b_i) < a\}$.

Так как W и X не пусты, то $a \neq 1$ и $v \neq b$. Любой лист $i \to j$, у которого $v \leqslant i < b$, начинается в вершине из X, поэтому должно выполняться $a \leqslant j$. Каждый лист $i \to j$, у которого $a < j < b$ и $b \to b_k \xrightarrow{*} i$, должен начинаться в некоторой вершине из $D(b_k)$, $k > k_0$. Отсюда вытекает, что $L_1(b_k) \geqslant a$. Но поскольку граф G 2-связен, то $L_1(v) < a$. Если бы вершина b не была первым потомком вершины v, то некоторый лист из X входил бы в вершину из W. Таким образом, вершина b должна быть первым потомком вершины v, а пара вершин $\{a, b\}$ — 2-сочленением 2-го типа, когда в определении 2-сочленения 2-го типа полагаем $r = v$. Следовательно, прямое утверждение леммы доказано. Заметим, что одно 2-сочленение может быть одновременно как 1-го, так и 2-го типов.

Лемма 2 дает простой критерий определения 2-сочленений графа G. Для нахождения пар 1-го типа мы рассматриваем каждую дугу $b \to v$ n-дерева P и проверяем, выполняется ли $L_2(v) \geqslant b$, а также одно из трех условий: не существует дуги $L_1(v) \to b$; $L_1(v) \neq 1$; вершина b имеет более одного сына. Если это так, то пара вершин $(L_1(v), b)$ является 2-сочленением 1-го типа. Для нахождения 2-сочленений 2-го типа нужно провести третий просмотр. Образуем новый список, состоящий из троек вершин (h, a, b); для краткости будем называть этот список 3-списком. То, что вершины образуют тройку, означает, что пара (a, b), возможно, является 2-сочленением 2-го типа и что h — самая высокая вершина, которая соединена с вершинами из $D(a) - D(b)$ маршрутом, не проходящим через вершины a и b.

(Здесь $h = H(b_{k_0+1})$, а вершина b_{k_0+1} была определена при доказательстве леммы 2.)

Просмотр глубины 1 выполняется так же, как осуществлялся второй просмотр; список троек образуется следующим образом: при прохождении листа (v, w) из начала 3-списка удаляются все тройки (h, a, b), у которых $w < a$. Если (h_1, a, b_1) — последняя удаленная тройка, то в 3-список добавляется новая тройка (h_1, w, b_1). Если же ни одна из троек не удаляется, то в 3-список добавляется тройка (v, w, v).

Когда бы мы ни возвращались во время просмотра к вершине $v \neq 1$ по дуге $v \to w$ n-дерева, для первой тройки (h, a, b) 3-списка проверяем, выполняется ли $v = a$. Если равенство выполняется, то пара вершин (a, b) является 2-сочленением 2-го типа. Удаляем также из начала 3-списка все тройки (h, a, b), у которых $h(w) > h$. Для вершины w, которая не является первым сыном вершины v, пусть $H(w)$ — наибольший (самый далекий) ее потомок. Удаляем теперь из начала 3-списка все тройки (h, a, b), у которых $H(w) \geqslant b$. Далее удаляем тройки, у которых $L_1(w) < a$. Если на последнем этапе тройки не удаляются и если нет дуги $L_1(w) \to v$, то в 3-список добавляется тройка $(H(w), L_1(w), v)$.

Если граф G имеет не менее одного 2-сочленения 2-го типа, то одно из них будет найдено, когда будет завершен третий просмотр.

Лемма 3. *Описанный выше метод позволяет найти 2-сочленение, если граф G не является 3-связным. Если же граф G 3-связен, то этим методом ни одна пара вершин не выделяется.*

Доказательство. Если граф G имеет 2-сочленение 1-го типа, то оно находится с помощью первой проверки, описанной выше; если в графе G нет 2-сочленений 1-го типа, то эта проверка не выделяет ни одной пары. Последнее вытекает из предложения (1) леммы 2, а вершина w, удовлетворяющая (1), существует тогда и только тогда, когда или нет дуги $L_1(v) \to b$, или $L_1(v) \neq 1$, или вершина b имеет более одного сына.

Предположим теперь, что в графе G нет 2-сочленений 1-го типа. Введем в рассмотрение проверку 2-го типа. Если тройка (h_1, a_1, b_1) находится в 3-списке выше тройки (h_2, a_2, b_2), причем $a_2 \leqslant a_1$, и если $a_2 = a_1$, то $b_2 \leqslant b_2$. Далее, если из 3-списка удаляется тройка (h, a, b) благодаря тому, что найден лист $v \to w$, у которого $w < a$, то $v < b$. Эти предложения можно доказать по индукции, используя упорядочение, которое определяется структурой смежности A. Для любой тройки (h, a, b) из 3-списка имеем $a \overset{*}{\to} b$, а также то, что вершина a является прямым потомком вершины, которая во время просмотра изучается в текущий момент.

Если рассматривается тройка (h, a, b) 3-списка и оказалось, что $v = a \neq 1$, когда возвращаемся по дуге $v \to w$ n-дерева, то простым доказательством по индукции получаем, что пара вершин (a, b) является 2-сочленением 2-го типа. Обратно, если $\{a, b\}$ — 2-сочленение 2-го типа, то положим $h = H(b_{k_0+1})$, где вершина b_{k_0+1} определена при доказательстве леммы 2. Пусть $a \to v \xrightarrow{*} b$, и пусть $i \to j$ — первый лист, который проходится в просмотре и удовлетворяет $v \leqslant i \leqslant h$. Тогда можно доказать опять по индукции, что тройка (i, j, i) находится в 3-списке, возможно модифицированном, и в конце концов выбирается как выделяющая 2-сочленение 2-го типа. Таким образом, выделение 2-сочленений 1-го и 2-го типов позволяет правильно определить, является ли граф G 3-связным.

Лемма 4. *Алгоритм определения 3-связности графа G требует $O(V, E)$ действий.*

Доказательство. На три просмотра, включая вспомогательные вычисления, требуется $O(V, E)$ действий. Если использовать основную сортировку, то на построение структуры смежности A затрачивается $O(V, E)$ действий. Выделение 2-сочленений 1-го типа требует $O(V)$ действий. Таким образом, все число действий, необходимое для выполнения алгоритма, линейно зависит от V и E.

3. ИЗОМОРФИЗМ ДЕРЕВЬЕВ

Пусть заданы два дерева T_1 и T_2; требуется определить, изоморфны ли они. Можем считать, что T_1 и T_2 — корневые деревья, поскольку в каждом некорневом дереве единственным образом выделяется вершина r_i, которую и назовем корнем. Эта вершина находится с помощью одновременного удаления всех вершин степени 1 (висячих вершин) дерева T_i и повторения такого удаления до тех пор, пока не останется одна вершина (*центр* дерева T_i) или одно ребро (концевые вершины этого ребра называются *бицентрами* дерева T_i). В последнем случае в середину указанного ребра помещаем новую вершину и получаем единственный корень. Число действий, необходимых для определения корня в каждом из деревьев, зависит линейно от числа вершин V, если тщательно реализовать процедуру выделения последней вершины или ребра.

Трудность установления изоморфизма деревьев заключается в том, что ребра, инцидентные произвольной вершине, не имеют фиксированного порядка. Если бы нам удалось преобразовать любое дерево к некоторому каноническому упорядоченному дереву, то проверка изоморфизма была бы простой — нужно проверить равенство канонических упорядоченных деревьев. По-

этому нам необходим алгоритм, который упорядочивает ребра, инцидентные каждой вершине дерева. Ряд авторов (Басакер и Саати [1965], Ледерберг [1964], Скойнс [1968], Вейнберг [1965]) предлагают такие способы упорядочения; эти способы мало отличаются друг от друга. Алгоритм Эдмондса (см. Басакер и Саати [1965]) является хорошим примером такого упорядочения. В данном корневом дереве вершины на каждом уровне упорядочиваются, начиная с вершин, которые расположены дальше всех от корня. После того как вершины на i-м уровне упорядочены, любой вершине $(i-1)$-го уровня приписывается список номеров ее сыновей (смежных с вершинами i-го уровня). Вершины на $(i-1)$-м уровне упорядочиваем лексикографически по их спискам в соответствии с порядком, который получили вершины на i-м уровне. После того как вершины упорядочены на каждом уровне, построить каноническое дерево легко. Но ни Эдмондс, ни другие авторы не заметили, что сам процесс упорядочения не так прост, и чтобы получить оценку числа действий $O(V)$, этот процесс нужно тщательно определить и аккуратно реализовать.

Для последующего полезно несколько обобщить задачу установления изоморфизма деревьев. Допустим, что каждой вершине приписано некоторое множество меток. Любая метка l — это целое число из отрезка $1 \leqslant l \leqslant V$, где V — число вершин дерева. Два помеченных дерева изоморфны, если можно установить взаимно однозначное соответствие между их непомеченными деревьями, такое, что любые две соответствующие вершины из разных помеченных деревьев имеют одно и то же множество меток. Теперь опишем алгоритм установления изоморфизма помеченных деревьев, который требует $O(V, L)$ действий, где V — число вершин, а L — число всех меток. Для сортировки применяется процедура основной сортировки или сортировки по ящикам, когда используется не более $2V+1$ ящиков. Алгоритм может быть запрограммирован на вычислительной машине с произвольным порядком выбора.

Если деревья некорневые, то в каждом из них с помощью описанной выше процедуры находится единственный корень. Для всех вершин определяются номера уровней. (Корень находится на уровне 0.) Далее, каждый раз как встречается метка l, строится упорядоченная пара (i, l), где i — номер уровня, на котором встречается эта метка. Используя две основные сортировки, упорядочиваем лексикографически множество построенных упорядоченных пар для того, чтобы определить список $\mathscr{L}_i$ меток, которые встречаются в каждом i-м уровне и которые записаны в порядке их появления.

Далее, применяем алгоритм Эдмондса, начиная с вершин самого высокого уровня и двигаясь к корню. Пусть k — уровень, рассматриваемый в текущий момент. Вершины $(k+1)$-го

уровня уже упорядочены, и им приписаны числа от 1 до N_{k+1}, где $N_{k+1} \leqslant V_{k+1}$, а V_{k+1} — число вершин $(k+1)$-го уровня. Используя список $\mathscr{L}_k$, изменяем метки k-го уровня. Наименьшую (по номеру) метку заменяем на $N_{k+1}+1$, следующую за ней — на $N_{k+1}+2$ и т. д. Затем для каждой вершины k-го уровня строим индексный список, в который записываются номера всех ее сыновей и все их метки. Числа в этих списках будут упорядочены, если списки строились следующим образом: Для каждого сына, имеющего номер i, выделяется элемент в индексном списке его отца. Такое выделение повторяется для каждого i из $1 \leqslant i \leqslant N_{k+1}$. Аналогичным образом вводятся номера меток. Сыновья (вершины $(k+1)$-го уровня) будут упорядочены благодаря тем действиям, которые выполнены на $(k+1)$-м уровне.

Теперь нужно упорядочить лексикографически вершины на k-м уровне по их индексным спискам. Каждое число n, которое встречается в индексном списке, преобразуется в упорядоченную пару (i, n), где i — номер позиции в индексном списке, в которой встречается это число. Опять, используя две основные сортировки, упорядочиваем лексикографически множество таких пар для того, чтобы получить список номеров (в соответствующем порядке), которые встречаются в каждой данной позиции i во всех индексных списках.

Пусть m — длина самого большого индексного списка. Применим теперь m основных сортировок, чтобы получить лексикографическое упорядочение индексных списков. В каждой из этих сортировок используется $N_{k+1}+|\mathscr{L}_k|$ ящиков. Во время i-го прохода к частично упорядоченному множеству вершин S добавляем вершины, индексные списки которых имеют длину $m-i+1$. Затем помещаем вершину $v \in S$ в ящики согласно величине $(m-i+1)$-го числа в индексном списке вершины v. Непустые ящики опустошаются в порядке, который определен списком P_{m-i+1}.

После m проходов вершины k-го уровня лексикографически упорядочены по их индексным спискам. Эти вершины получили номера от 1 до N_k для некоторого числа N_k, не превышающего V_k; двум вершинам приписан один и тот же номер, если их индексные списки совпадают. Этим завершается процесс на k-м уровне. После того как вершины на каждом уровне упорядочены, легко построить канонические упорядоченные деревья, после чего установление изоморфизма сводится к определению простого равенства.

Нахождение корня дерева осуществляется за $O(V)$ действий. Построение списков меток $\mathscr{L}_i$ требует $O(L)$ действий, а на построение индексных списков k-го уровня тратится $O(V_{k+1}+|\mathscr{L}_k|)$ действий. Лексикографическое упорядочение вершин на k-м уровне по их индексным спискам требует m проходов и только

$O(V_{k+1} + |\mathscr{L}_k|)$ действий, поскольку на каждом проходе опустошаются только непустные ящики, а полное число всех размещений вершин в ящики равно величине $V_{k+1} + |\mathscr{L}_k|$. Но $\sum_n (V_{k+1} + |\mathscr{L}_k|) \leqslant V + L$, поэтому число действий алгоритма в целом ограничено $O(V, L)$. Если деревья не помечены или если каждое дерево имеет самое большее одну пометку, то для установления изоморфизма таких деревьев требуется $O(V)$ действий.

4. ИЗОМОРФИЗМ 3-СВЯЗНЫХ ПЛАНАРНЫХ ГРАФОВ

В этом разделе описывается алгоритм разбиения всех 3-связных графов на подмножества изоморфных графов. Асимптотически число действий алгоритма пропорционально $V \cdot \log V$, где V — число вершин в графах. Аналогичный алгоритм такого разбиения, основанный на преобразовании графов в конечные автоматы, был ранее разработан Хопкрофтом [1971]. Однако переход к графам приводит к существенным упрощениям, поскольку конечные автоматы, которые выделяются при преобразовании только планарных графов, образуют очень небольшое подмножество множества всех конечных графов, и нет нужды находить разбиение всех конечных автоматов.

Пусть G — планарный граф, компонентами связности которого являются 3-связные графы, отличные от n-угольников и n-связок. Рассмотрим конкретное расположение (вложение) графа G на плоскости. Каждое ребро графа G будем рассматривать как две дуги, имеющие противоположные ориентации. Пусть (v_1, v_2) — одна из таких дуг. Дугу (v_2, v_1) будем обозначать через $(v_1, v_2)^r$. Введем обозначения $(v_1, v_2) \vdash\!\!\!\overset{R}{-} (v_2, v_3)$ и $(v_1, v_2) \vdash\!\!\!\overset{L}{-} (v_2, v_4)$, где (v_2, v_3) и (v_2, v_4) — дуги, которые ограничивают грани соответственно справа и слева от дуги (v_1, v_2).

Обозначим через ε последовательность нулевой длины. Для каждой дуги e будем писать $e \vdash\!\!\!\overset{\varepsilon}{-} e$. Пусть x — последовательность символов R и L. Если $e_1 \vdash\!\!\!\overset{x}{-} e_2$ и $e_2 \vdash\!\!\!\overset{R}{-} e_3$ (или $e_2 \vdash\!\!\!\overset{L}{-} e_3$), то будем писать $e_1 \vdash\!\!\!\overset{xR}{-} e_3$ (или соответственно $e_1 \vdash\!\!\!\overset{xL}{-} e_3$). Будем использовать также обозначение $e_1 \vdash e_2$, когда ясно, какова последовательность x. Интуитивно мы используем запись $e_1 \vdash\!\!\!\overset{x}{-} e_2$ тогда, когда дуга e_2 достижима из дуги e_1, если идти по тому пути (графа), который определяется последовательностью x. Каждый символ из x указывает, в какую сторону нужно повернуть, входя в вершину. Если в вершину v входим по дуге e, то выходим из v по дуге, которая расположена непосредственно справа или слева от дуги e в зависимости от того, какой символ (соответственно R или L) указан в последовательности x. Заметим, что из $e_1 \overset{*}{\Rightarrow} e_2$ не обязательно следует $e_1 \vdash e_2$, поскольку

$e_1 \overset{*}{\Rightarrow} e_2$ обозначает произвольный путь, а $e_1 \vdash e_2$ — обозначение специального пути. Пусть λ — такое отображение множества дуг в множество целых чисел, что $\lambda(e_1) = \lambda(e_2)$ в том и только том случае, когда число дуг на грани с правой (левой) стороны от дуги e_1 равно числу дуг на грани с правой (левой) стороны от дуги e_2, а также когда степени вершин, являющихся началами дуг e_1, e_2, совпадают и степени вершин графа G, являющихся концами дуг e_1, e_2, также совпадают.

Лемма 5. *Пусть e — произвольная дуга, а v — вершина из той компоненты связности, которая содержит e. Тогда*

1) *существует дуга e_1, ориентированная к вершине v, такая, что $e \vdash e_1$;*

2) *если e_1 — дуга, ориентированная к вершине v, и если вершина v имеет нечетную степень, то $e \vdash e_1$, если же степень вершины v четна, то или $e \vdash e_1$, или $e \vdash e_1^r$.*

Доказательство. Для доказательства первого предложения достаточно показать, что для любой вершины v, смежной с концом дуги e, существует дуга e', ориентированная к вершине v, такая, что $e \vdash e'$. Пусть $e_1, e_2, \ldots, e_m$ — дуги, ориентированные к вершине, являющейся концом дуги e. Рассмотрим путь, состоящий из дуг e_1, e_2^r, после которых идут дуги вокруг грани с левой стороны от дуги e_2^r до дуги e_3, затем идет дуга e_4^r, после которой идут дуги вокруг грани слева от дуги e_4 и так далее, возвращаясь к дуге e_1. Этот путь проходит через каждую вершину, смежную с вершиной v.

Следуя по пути к вершине v и проходя затем дважды вокруг вершины v по пути подобно тому, как это было сделано при доказательстве предложения 1, в результате получаем предложение 2.

Лемма 6. *Пусть e_1, e_2, e_3, e_4 — дуги. Тогда*

1) *если $e_1 \vdash e_2$, то $e_2 \vdash e_1$ и $e_1^r \vdash e_2^r$;*

2) *если $e_1 \vdash e_1^r$, то $e_1 \vdash e_2$;*

3) *если существуют путь p_1: $e_1 \overset{*}{\Rightarrow} e_3$ и соответствующий путь p_2: $e_2 \overset{*}{\Rightarrow} e_4$ (путь p_2 соответствует пути p_1, если оба пути имеют одинаковую длину и если одинаковы числа дуг, подсчитываемых по часовой стрелке между парами дуг, входящих и выходящих из соответствующих вершин путей, причем соответствующие дуги имеют одно и то же значение λ), то или $e_1 \vdash e_3$ и $e_2 \vdash e_4$, или $e_1 \vdash e_3^r$ и $e_2 \vdash e_4^r$ на одной и той же последовательности правых и левых поворотов.*

Доказательство. 1) Рассмотрим путь p из e_1 к e_2. Пусть e — предпоследняя дуга в пути p. Не теряя общности, можно

предположить, что дуга e_2 ограничивает грань с правой стороны от e, и нужно только показать, что $e_2 \vdash e$. Ясно, что соединение $e_2 \vdash e$ реализуется путем, который проходит по часовой стрелке вокруг грани справа от дуги e_2. Установив, что $e_2 \vdash e_1$, немедленно получаем $e_1^r \vdash e_2^r$, если изменить ориентации всех дуг пути $e_2 \vdash e_1$.

2) Из леммы 5 следует или $e_1 \vdash e_2$, или $e_1 \vdash e_2^r$. Если же $e_1 \vdash e_2^r$, то в силу предложения 1 настоящей леммы имеем $e_1^r \vdash e_2$.

3) Из построения леммы 5 вытекает, что или $e_1 \vdash e_3$, или $e_1 \vdash e_3^r$ реализуется путем, который содержит только дуги, ограничивающие грани, смежные с p_1. Соответствующее построение, в котором используется путь p_2, приводит к необходимому результату.

Две дуги e_1 и e_2 называются *различимыми* в том и только том случае, когда существует такая последовательность x, что $e_1 |\overset{x}{-} e_3$, $e_2 |\overset{x}{-} e_4$ и $\lambda(e_3) \neq \lambda(e_4)$. Если дуги e_1 и e_2 не являются различимыми, то будем называть их *неразличимыми*.

Нам потребуется следующая вспомогательная лемма.

Лемма 7. *Пусть G — 2-связный планарный граф, а $(v_1, v_2)(v_2, v_3), \ldots, (v_{n-1}, v_n)$ — простой путь в G. Тогда существует грань, имеющая общую дугу с этим путем и обладающая следующим свойством: множество дуг, принадлежащих как грани, так и пути, образуют непрерывный сегмент пути* [1]). *Более того, при проходе по дуге грани в движении по пути от v_1 к v_n грань остается с правой стороны.*

Доказательство см. в (Хопкрофт [1971 А]).

Теорема 8. *Дуги e и e' неразличимы тогда и только тогда, когда существует изоморфизм вложенного графа G, который отображает e на e'.*

Доказательство. Достаточность условия очевидна. Именно, если дуга e отображается на дугу e' некоторым изоморфизмом, то, как легко видеть, эти дуги неразличимы. Необходимость доказать труднее; докажем сначала необходимость условия для регулярных графов степени 3.

Пусть G — регулярный граф степени 3, а e и e' — неразличимые дуги. Покажем теперь, как построить изоморфизм, который отождествляет дуги e и e'. Если e и e' принадлежат одной и той же компоненте связности, то любая дуга, не принадлежащая этой компоненте, отображается на себя. Если дуги e и e'

[1]) То есть общие дуги в рассматриваемом пути образуют также путь. — *Прим. перев.*

принадлежат различным компонентам, скажем C_1 и C_2, то каждая дуга, не принадлежащая ни C_1, ни C_2, также отображается на себя.

Отождествим теперь дугу e с дугой e'. Дуги, обратные дугам e, e', также отождествляются; автоматически отождествляются вершины, являющиеся началами и концами этих дуг. Если дуга e_1 отождествляется с дугой e_2, то производится отождествление дуг e_3, e_4, где $e_1 |\overset{R}{-} e_3$ и $e_2 |\overset{R}{-} e_4$. Если дуга e_3 была уже отождествлена с дугой e_4, то выбираем некоторую пару дуг e_5, e_6, которая также была отождествлена, так что дуги e_7, e_8 не отождествлены и $e_5 |\overset{L}{-} e_7$, а $e_6 |\overset{L}{-} e_8$. Далее производим отождествление дуг e_7, e_8, и этот процесс повторяется. Другими словами, для получения новых дуг всегда используем символ R (если возможно такое получение), в ином случае используем символ L. Проведение процесса означает, что мы всегда отождествляем дуги, идя по некоторому пути до тех пор, пока не будет достигнута пара вершин, которая уже была отождествлена.

В силу предложения 2 леммы 5, если не возникает противоречие, то указанная выше процедура позволяет построить необходимый изоморфизм. Противоречие возникает при попытке отождествить вершину v_1 с вершиной v_2, которая была отождествлена с некоторой вершиной v_3, отличной от v_1. Докажем теперь, что такая ситуация невозможна.

Предположим противное — противоречия возможны, и рассмотрим первый случай наступления такого противоречия. Одна из дуг в последней отождествленной паре дуг должна замыкать некоторый простой цикл. Важно отметить, что дуга, замыкающая цикл, должна кончаться вершиной, которая инцидентна двум другим уже отождествленным дугам. Соответствующая дуга или не замыкает цикл (т. е. ее концом является неотождествленная вершина), или замыкает другой цикл (концевые вершины двух дуг ранее были отождествлены, но не друг с другом). В последнем случае циклы имеют различные длины. Если обе дуги замыкают циклы, то обозначим через c цикл, имеющий меньшую длину. Если замыкается только один цикл, то обозначим его также через c. Пусть p — путь во втором графе, который соответствует вершинам цикла c первого графа. Первой и последней вершине пути p соответствует одна вершина цикла c.

Поскольку имеется цикл, который отображается в простой путь, то можем выбрать цикл c, который отображается в простой путь, но любой другой цикл, состоящий из вершин цикла c и вершин внутри цикла c, не отображается в простой путь. Из леммы 7 вытекает, что некоторая грань смежна с путем p справа и все дуги этой грани, принадлежащие и пути p, образуют в p непрерывный сегмент. Начинаем отождествлять дуги этой гра-

ни с дугами внутри цикла c. Здесь возможны три случая: 1) возвращаемся в вершину пути p раньше, чем в вершину цикла c; 2) возвращаемся в вершину цикла c раньше, чем в вершину пути p; 3) первый и второй случаи наступают одновременно. В первом случае грань отождествляется с незамкнутым путем. Это невозможно, так как отображение λ содержит информацию о числе дуг в грани как справа, так и слева от произвольной выделенной дуги. Второй случай также невозможен, поскольку ни один из циклов внутри цикла c не отображаются в незамкнутый путь. В случае 3 нужно отождествить соответствующие грани. Тогда пути оканчиваются в соответствующих вершинах и цикл c разбивается на два цикла c_1 и c_2. Предположим, что цикл c_1 соответствует грани. Тогда цикл c_2 отображается в путь; пришли к противоречию. Поскольку в каждом случае пришли к противоречию, то вынуждены заключить, что высказанное выше предположение невозможно.

Установив утверждение теоремы для частного случая, для регулярных графов степени 3, докажем теперь теорему в общем случае. Пусть G — укладка планарного графа, все компоненты связности которого 3-связны. Предположим, что дуги e_1 и e_2 неразличимы, но не существует изоморфизм, отображающий дугу e_1 на дугу e_2. Обозначим через $\hat{G}$ граф, полученный из G с помощью замены каждой вершины степени $d > 3$ на d-угольник. Пусть $\hat{e}_1$, $\hat{e}_2$ — дуги графа $\hat{C}$, соответствующие дугам e_1, e_2. Поскольку $\hat{G}$ — регулярный граф степени 3, то дуги $\hat{e}_1$ и $\hat{e}_2$ должны быть различимыми. Пусть $\hat{p}$ — путь, различающий e_1 и $\hat{e}_2$. Ясно, что в G существует соответствующий путь. Из леммы 6 (предложение 3) следует, что существует последовательность x, которая различает e_1 и e_2. Пришли к противоречию. Теорема доказана.

Алгоритм установления изоморфизма за $O(V \cdot \log V)$ действий зависит от эффективного алгоритма разбиения множества дуг планарного графа на подмножества таким образом, что две дуги принадлежат одному и тому же множеству тогда и только тогда, когда эти дуги неразличимы. Разбиение отыскивается следующим образом.

Сначала дуги разбиваются так, что дуги e_1 и e_2 относим к одному множеству в том и только том случае, когда $\lambda(e_1) = \lambda(e_2)$. Номер каждого множества разбиения, за исключением одного, заносим в два списка, один из них называется правым списком (для краткости R-списком), а другой — левым списком (для краткости L-списком). Пусть в данный момент имеются множества разбиения $B_1, B_2, \ldots, B_i$. Множества разбиения изменяются с помощью предлагаемой ниже процедуры до тех пор, пока не будут опустошены R- и L-списки.

Выберем номер j некоторого множества из R- или L-списков и удалим его из этого списка. Предположим, что j входило в R-список. Для каждой дуги e из множества B_j отмечаем дугу e', которая определяется с помощью $e \overset{R}{\vdash} e'$. Если множество B_i содержит как отмеченные, так и неотмеченные дуги, то разбиваем его на два подмножества B_{i_1} и B_{i_2}, так что одно из них состоит только из отмеченных дуг, а другое — только из неотмеченных дуг. Пусть $|B_{i_1}| \leqslant |B_{i_2}|$. Если номер i уже содержится в R-списке, то заменяем его на номера i_1 и i_2, если не содержится, то добавляем в R-список только i_1. Аналогично если номер i содержался в L-списке, то заменяем его на i_1 и i_2, если не содержался, то в L-список помещаем только i_1.

Теорема 9. *Описанный выше алгоритм конечен, и в конце его работы две дуги принадлежат одному и тому же множеству разбиения тогда и только тогда, когда эти дуги неразличимы.*

Доказательство. Достаточность. В начале работы алгоритма дуги размещаются в различные множества, если они различаются сразу. Затем множество разбивается так, что пара дуг e_1, e_2 переходит в разные множества, если существуют дуги e_3, e_4, принадлежащие уже различным множествам, такие, что $e_3 \overset{R}{\vdash} e_1$ и $e_4 \overset{R}{\vdash} e_2$ или $e_3 \overset{L}{\vdash} e_1$ и $e_4 \overset{L}{\vdash} e_1$. В первом случае $e_1^r \overset{L}{\vdash} e_3^r$ и $e_2^r \overset{L}{\vdash} e_4^r$, во втором случае $e_1^r \overset{R}{\vdash} e_3^r$ и $e_2^r \overset{L}{\vdash} e_4^r$. В любом случае дуги e_1 и e_2 различимы.

Необходимость. Предположим, что дуги e_1 и e_2 различимы. Индукцией по длине n наименьшей последовательности, различающей дуги e_1^r и e_2^r, докажем, что дуги e_1 и e_2 отнесены к разным множествам. Предположим, что индуктивное предположение справедливо для последовательностей длины n и что длина наименьшей последовательности, различающей дуги e_1^r и e_2^r, равна $n+1$. Не теряя общности, можно считать, что этой последовательностью является xR. Тогда последовательность x имеет длину n и различает некоторую пару дуг e_3^r и e_4^r, где $e_3 \overset{R}{\vdash} e_1$ и $e_4 \overset{R}{\vdash} e_2$. В силу индуктивного предположения дуги e_3, e_4 относятся к различным множествам. Номер одного из этих множеств помещается в R-список, и после его удаления дуги e_1 и e_2 должны быть отнесены к разным множествам разбиения.

Теорема 10. *Число действий алгоритма разбиения ограничено величиной $k \cdot V \cdot \log V$, где k — некоторая константа.*

Доказательство. Ясно, что число действий алгоритма не меньше числа действий, затрачиваемых на нахождение разбиения. Предположим, что в некоторый момент $B_1, B_2, \ldots, B_m$ — множества разбиения. Обозначим через b_i число дуг множества

B_i. Введем также обозначения: M — множество натуральных чисел 1, 2, ..., m; I — множество номеров R-списка; J — множество номеров L-списка. Покажем, что число действий, оставшихся до окончания работы алгоритма, ограничено сверху величиной

$$T = k \cdot \Big(\sum_{i \in I} b_i \log b_i + \sum_{i \in M \setminus I} b_i \log (b_i/2) + \sum_{i \in J} b_i \log b_i + \\ + \sum_{i \in M \setminus J} b_i \log (b_i/2)\Big).$$

Ясно, что эта оценка выполняется, когда алгоритм кончает работу. Посмотрим, что произойдет, если номер i_0 выбран из множества I. Некоторые множества будут разбиты. Число действий до окончания работы алгоритма будет ограничено новой величиной T'. Число действий, затрачиваемых на разбиение, оценивается $k \cdot b_{i_0}$. Нам нужно показать, что

$$k \cdot b_{i_0} + T' \leqslant T.$$

Предположим, что множество, содержащее b_i дуг, разбивается на подмножества, содержащие c_i и $b_i - c_i$ дуг, где $c_i \leqslant b_i/2$. Имеем

$$b_i \cdot \log b_i \geqslant c_i \cdot \log c_i + (b_i - c_i) \cdot \log (b_i - c_i),$$
$$b_i \cdot \log b_i/2 \geqslant c_i \cdot \log c_i/2 + (b_i - c_i) \cdot \log (b_i - c_i)/2.$$

Таким образом, нам нужно только показать, что

$$k \cdot b_{i_0} + k \cdot b_{i_0} \cdot \log (b_{i_0}/2) \leqslant k \cdot b_{i_0} \cdot \log b_{i_0}.$$

Последнее выполняется в силу того, что

$$b_{i_0} + b_{i_0} \cdot \log b_{i_0}/2 = b_{i_0} (1 + \log b_{i_0}/2) = b_{i_0} \cdot \log b_{i_0}.$$

Этим доказательство теоремы заканчивается.

Описанный алгоритм разбиения можно применить для получения разбиения множества 3-связных планарных графов на подмножества изоморфных графов. Любой 3-связный планарный граф имеет в точности две укладки на плоскости. Одна укладка получается из другой с помощью одновременного изменения порядка всех дуг, инцидентных каждой вершине, на обратный порядок. Сопоставим этим укладкам направление: одна укладка — по часовой стрелке, другая — против часовой стрелки. Для данного набора G_1, G_2, ..., G_n 3-связных планарных графов образуем составной граф G, который состоит из пар копий всех графов $G_i (1 \leqslant i \leqslant n)$. Граф G_i вкладывается в плоскость таким образом, чтобы одна копия каждого графа имела укладку по часовой стрелке, а другая копия этого графа имела укладку против часовой стрелки. Теперь к укладке графа G применяется алгоритм разбиения. Из теорем 8 и 9 следует, что граф G_i

изоморфен графу G_j тогда и только тогда, когда для любой дуги e укладки графа G_i существует дуга e' в одной из укладок графа G_j, такая, что дуги e, e' принадлежат одному и тому же множеству разбиения всех дуг. Теперь легко разбить составной граф G на изоморфные графы G_i, именно отнесем граф G_1 в первое множество разбиения. Выберем некоторую дугу e графа G_1. Просматриваем множество разбиения, содержащее дугу e. Для каждого j, такого, что любая укладка графа G_j имеет дугу из этого же множества разбиения, размещаем G_j в множество, которое содержит G_1. Выберем наименьшее k, такое, что граф G_k не помещен в множество 1, и отнесем граф G_k к множеству 2. Далее выбираем в G_k некоторую дугу e и так же, как и раньше, просматриваем то множество разбиения, которое содержит дугу e. Процесс повторяется до тех пор, пока каждый граф G_k не будет отнесен к некоторому множеству.

Число действий, необходимое для завершения описанного выше процесса, оценивается следующим образом: сначала определяются планарные укладки графов G_i. Для каждого графа G_i необходимое число действий пропорционально числу вершин графа G_i (см. Тарьян [1972 В]). Таким образом, на этом этапе общее число действий пропорционально числу вершин составного графа G. Число действий, затрачиваемое на нахождение разбиения дуг, оценивается величиной $k \cdot E \cdot \log E$, где k — некоторая константа, а E — число дуг составного графа G. Следовательно, число действий в целом ограничивается величиной $E \cdot \log E$, умноженной на некоторую константу. Поскольку в планарном графе $E \leqslant 3V - 6$, то последней оценкой является величина $O(V \cdot \log V)$, где V — число вершин графа G.

5. АЛГОРИТМ УСТАНОВЛЕНИЯ ИЗОМОРФИЗМА

Пусть даны два графа G_1 и G_2. В алгоритме определения изоморфизма этих графов в каждом из них находятся компоненты связности, в каждой компоненте связности выделяются 2-связные компоненты и в каждой 2-связной компоненте определяются 3-связные компоненты. Структура связности любого графа представляется следующим образом: для каждого графа определяется дерево, состоящее из корня и вершин, представляющих компоненты связности. Для каждой компоненты связности определяется дерево, в котором одной вершиной представляется каждая 2-связная компонента и также одной вершиной представляется каждая точка сочленения. Пусть v_a — вершина, соответствующая точке сочленения a, а v_B — вершина, соответствующая 2-связной компоненте B, которая содержит вершину a. Тогда вершины v_a и v_B соединяются ребром. Легко видеть, что определенный таким образом граф является деревом. Ребра этого де-

рева, представляющего компоненту связности, назовем 2-*лепестками*.

Каждую 2-связную компоненту также представляем деревом. В множестве его вершин одна вершина представляет каждое 2-сочленение и одна вершина представляет каждую 3-связную компоненту рассматриваемого графа. Если вершина v_c представляет 3-связную компоненту C, а вершина v_{ab} — 2-сочленение (a, b), то проводим ребро, соединяющее v_c и v_{ab}, если и только если обе вершины a, b принадлежат C. Ребра дерева, представляющие 3-связную компоненту, назовем 3-*лепестками*.

Рассмотрим деревья, соответствующие компонентам связности. Каждый 2-лепесток есть 2-связная компонента, которой в свою очередь соответствует структура типа дерева 3-связных компонент. Всем 3-связным компонентам, которые являются 2-листьями и которые содержатся в 2-листьях, приписывается упорядоченная пара числовых кодов, таких, что равенство этих кодов означает изоморфизм. Последнее осуществляется с помощью методики, описанной ниже. Выделенные 3-связные компоненты удаляются, а их коды приписываются новым ребрам, которые соединяют в остающейся части графа 2-сочленения, содержащиеся в удаленных 3-связных компонентах. Этот процесс создает новые 3-лепестки в имеющихся 2-лепестках. Находим новые 3-лепестки и для них определяем коды. Процесс повторяется до тех пор, пока каждая 2-связная компонента, которая является 2-лепестком, не сведется к одному ребру. Далее такие ребра удаляются, а их коды приписываются соответствующим точкам сочленения в оставшемся графе. Опять находим новые 2-связные компоненты, являющиеся 2-лепестками, и процесс повторяется. В результате каждая компонента связности будет сведена к единственной вершине, которой приписан код, который необходим для установления изоморфизма. Коды компонент связности каждого графа упорядочиваются. Упорядоченные коды разных графов сравниваются на совпадение. Если коды равны (совпадают), то графы изоморфны, если не равны, то графы не изоморфны.

Рассмотрим произвольный 3-лепесток. Отметим, что он имеет ориентацию относительно его 2-сочленений. В силу этого обстоятельства 3-лепестку приписывается именно упорядоченная пара чисел. Числа равны тогда и только тогда, когда 3-лепесток симметричен относительно замены его 2-сочленений. Для того чтобы приписать пары целых чисел множеству 3-лепестков, устанавливаем сначала планарность компонент с помощью алгоритма, которые характеризуются линейным числом действий (см. Хопкрофт и Тарьян [1972A], Тарьян [1972B]). Если хотя бы одна компонента не является планарным графом, то алгоритм установления изоморфизма графов «не работает», поскольку и весь граф не планарен. Если же все 3-лепестки планарны, то

алгоритм определения планарности строит планарное представление компоненты. В силу 3-связности компонента определяется единственным образом. Алгоритм, описанный в разд. 4 и характеризующийся числом действий $O(V \cdot \log V)$, используется далее для нахождения классов эквивалентности изоморфных 3-лепестков. 2-сочленениям каждого 3-лепестка в соответствии с классами эквивалентности приписываем специальные целочисленные коды таким образом, чтобы изоморфным компонентам приписывались одинаковые упорядоченные пары целых чисел, а неизоморфным компонентам приписывались разные упорядоченные пары.

Число действий алгоритма оценивается числом действий, требуемых для нахождения разбиения 3-связных компонент на классы (эквивалентности) изоморфных графов. Если V — количество всех вершин в каждом из исходных графов, то на получение разбиения затрачивается $O(V \cdot \log V)$ действий. Число действий, затрачиваемых на нахождение 2-связных и 3-связных компонент, на определение планарности всех 3-лепестков и на построение планарного представления, является величиной $O(V)$, которая меньше, чем $O(V \cdot \log V)$.

ЛИТЕРАТУРА

Ariyoshi A., Shirakawa I., Hiroshi O. [1971] Decomposition of a graph into compactly connected two-terminal subgraphs, IEEE Transactions on Circuit Theory, Vol. CT-18, No. 4, 430—435.

Busacker R. G., Saaty T. L. [1965] Finite Graphs and Networks: An Introduction with Applications, McGraw-Hill, New York, 196—199 [русский перевод: Р. Басакер, Т. Саати, Конечные графы и сети, М., «Наука», 1974].

Edmonds J. [1965] Paths, trees and flowers, 2 Canad. J. Math., 17, 449—467.

Harary F. [1969] Graph Theory, Addison-Wesley, Reading, Massachusetts [русский перевод: Ф. Харари, Теория графов, М., «Мир», 1973].

Hopcroft J. [1971A] An $N \log N$ algorithm for isomorphism of planar triply connected graphs, Tecnical Report STAN-CS-71-192, Computer Science Department, Stanford University, Stanford, California.

[1971B] A $N \log N$ algorithm for minimizing states in a finite automaton, Theory of Machines and Computations, Academic Press, New York, 189—196.

Hopcroft J., Tarjan R. [1971A] A V^2 algorithm for determining isomorphism of planar graphs, Information Processing Letters, Vol. I, North Holland Publishing Company, Amsterdam, 32—34.

[1971B] Efficient algorithms for graph manipulation. Tecnical Report STAN-CS-71-207, Computer Science Department, Stanford University (to appear in C ACM).

[1971C] Planarity testing in $V \log V$ steps, Information Processing 1971, Proceedings of IFIP Congress 71.

[1974] Efficient Planarity Testing, Journal of ACM, 21, No. 4, 549—569.

Lederberg J. [1964] Dendral-64: A system for computer construction, enumeration and notation of organic molecules as tree structure and cyclic graphs Part I: Notational algorithm for tree structures, Interim Report to the National Aeronautics and Space Administration, Grant NSG 81-60, NASA Cr 68898, STAR No. N-66-14074.

Scoins H. I. [1968] Placing trees in lexicongraphic order, Machine Intelligence 3, American Elsevier Publishing Co., New York, 43—60.
Tarjan R. [1971] Depth-first search and linear graph algorithms, Twelfth Annual Symposium on Switching and Automata Theory, IEEE, 114—119.
[1972] An efficient planarity algorithm, Ph. D. Thesis. Tecnical Report STAN-CS-244-71, Computer Science Department, Stanford University, Stanford, California.
[1974] Finding dominators in directed graphs, *SIAM J. Comput.*, **3**, No. 1, 62—89.
Tutte W. T. [1966] Connectivity in Graphs, Oxford Press, London.
Weinberg L. [1965] Plane representations and codes for planar graphs, Proceedings of the Third Annual Allerton Conference on Circuit and System Theory, 733—744.

[1]) Статьи, отмеченные значком *, добавлены при переводе. В статье Тарьяна демонстрируется еще одно применение используемой авторами методики. — *Прим. перев.*

Слабая сингулярная теория второго порядка функции следования не элементарно рекурсивна[1])

А. Р. Мейер

Пусть L_{S1S} — множество формул, выразимых в слабой сингулярной логике второго порядка, использующей лишь предикаты $[x = y + 1]$ и $[x \in X]$. Бюхи показал, что множество истинных предложений в L_{S1S} (относительно стандартной интерпретации $\langle \mathbb{N}$, функция следования$\rangle$ с переменными второго порядка, интерпретируемыми в области конечных множеств) является рекурсивным. Через $WS1S$ мы обозначаем множество истинных предложений из L_{S1S}. Мы докажем, что $WS1S$ не является элементарно рекурсивным в смысле Кальмара. В действительности мы утверждаем более сильный результат.

Теорема 1. *Существует константа* $\varepsilon > 0$, *такая, что если* $\mathfrak{M}$ — *машина Тьюринга, которая, будучи запущена с произвольным предложением из* L_{S1S} *на ленте, останавливается в указанном допускающем состоянии тогда и только тогда, когда предложение истинно, то для всех достаточно больших* n *существует предложение длины* n, *для которого вычисление на* $\mathfrak{M}$ *требует*

$$2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \Big\} [\varepsilon \cdot \log_2 n]$$

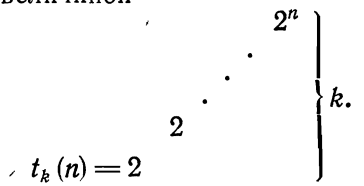
шагов и клеток ленты.

Пусть $t_0(n) = n$, $t_{k+1}(n) = 2^{t_k(n)}$. Хорошо известная характеристика элементарно рекурсивных функций, данная Р. В. Ричи, показывает, что множество предложений является элементарно рекурсивным тогда и только тогда, когда оно распознаваемо на ленте, ограниченной при некотором фиксированном k и всех вхо-

---

[1]) Meyer A. R., Weak monadic second order theory of successor is not elementary-recursive. Preliminary Report, 1973.

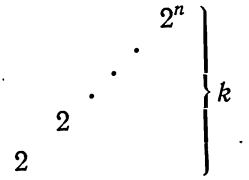
дах длины $n \geqslant 0$ величиной



Следовательно, $WS1S$ не элементарно рекурсивно.

В этих заметках мы доказываем несколько менее сильный вариант теоремы 1, который, согласно результату Ричи, является все еще достаточным, чтобы установить истинность нашего заголовка.

Теорема 2. *Пусть* $\mathfrak{M}$ *— машина Тьюринга, которая, будучи запущена с произвольным предложением из* L_{S1S} *на ленте, останавливается в указанном допускающем состоянии тогда и только тогда, когда предложение истинно. Тогда для всякого* $k \geqslant 0$ *существует бесконечно много n, для которых вычисление на* $\mathfrak{M}$ *требует для некоторого предложения длины n*



шагов и клеток ленты.

В основе нашего доказательства лежит идея — показать, что существуют предложения из L_{S1S}, которые описывают вычисления на машинах Тьюринга при условии, что зона, необходимая для вычисления, не больше, чем t_k (размер предложения). Так как машина Тьюринга, использующая данную зону, может моделировать любую машину Тьюринга, использующую меньшую зону, мы заключаем, что короткие предложения из L_{S1S} могут описывать неустранимо длинные вычисления и, следовательно, само L_{S1S} должно быть трудно разрешимым.

В действительности удобно будет развить вспомогательные обозначения для множеств конечных последовательностей, которые мы называем γ-выражениями. Мы покажем, что γ-выражения могут описывать в подходящем смысле вычисления на машинах Тьюринга и что L_{S1S} может описывать свойства γ-выражений.

Определение. Пусть Σ — конечное множество, элементы которого называются символами. Σ^* есть множество всех конечных последовательностей символов из Σ. Для $x, y \in \Sigma^*$ конка-

тенация x и y, записываемая как $x \cdot y$ или xy, есть последовательность, состоящая из символов x и следующих за ними символов y. Элемент $x \in \Sigma^*$ называется словом. Длина x обозначается $l(x)$. Мы используем λ для обозначения пустой последовательности из Σ^* длины нуль, которая, по определению, обладает свойством: для всякого $x \in \Sigma^*$ $x \cdot \lambda = \lambda \cdot x = x$ (Σ^* есть свободный моноид, порожденный Σ, с единицей λ). Конкатенация распространяется на подмножества A, $B \subset \Sigma^*$ с помощью правила

$$A \cdot B = AB = \{xy \mid x \in A,\ y \in B\}.$$

Для всякого $A \subset \Sigma^*$ мы определяем

$$A^0 = \{\lambda\}, \quad A^{n+1} = A^n \cdot A, \quad A^* = \bigcup_{n=0}^{\infty} A^n.$$

Эти операции хорошо известны в теории автоматов. Далее мы вводим одно отображение.

Определение. Для любого Σ функция γ_Σ: $P(\Sigma^*) \to P(\Sigma^*)$ определяется с помощью правил [1]):

$$\gamma_\Sigma(\{x\}) = \{y \in \Sigma^* \mid l(y) = l(x)\} = \Sigma^{l(x)} \quad \text{для} \quad x \in \Sigma^*,$$

$$\gamma_\Sigma(A) = \bigcup_{x \in A} \gamma(\{x\}) \quad \text{для} \quad A \subset \Sigma^*.$$

Мы опускаем нижний индекс у γ_Σ, когда Σ ясно из контекста.

γ-выражения над Σ суть слова в $\Sigma \cup \{\gamma, \cdot, \neg, \cup, (,)\}$, где γ, $\cdot$, $\neg$, $\cup$, $(,)$ — символы, *не принадлежащие* Σ. Каждое γ-выражение α определяет множество $L(\alpha) \subset \Sigma^*$.

Определение. Для всякого Σ γ-выражения над Σ и функция L: {γ-выражения над Σ} $\to P(\Sigma^*)$ определяются индуктивно следующим образом:

1) σ есть γ-выражение над Σ для любого $\sigma \in \Sigma$ и $L(\sigma) = \{\sigma\}$;

2) $(\alpha \cdot \beta)$, $(\alpha \cup \beta)$, $\neg(\alpha)$ и $\gamma(\alpha)$ суть γ-выражения над Σ [2]) и $L((\alpha \cdot \beta)) = L(\alpha) \cdot L(\beta)$, $L((\alpha \cup \beta)) = L(\alpha) \cup L(\beta)$, $L(\neg(\alpha)) = \Sigma^* - L(\alpha)$, $L(\gamma(\alpha)) = \gamma(L(\alpha))$.

Пояснив таким образом различие между γ-выражением α и множеством $L(\alpha)$, которое оно определяет, мы часто будем игнорировать это различие, если не возникает недоразумение. Так, мы пишем $\Sigma^* = \sigma \cup \neg(\sigma)$ вместо $\Sigma^* = L((\sigma \cup \neg(\sigma)))$. Подобным образом для любого множества букв $V \subset \Sigma$

$$V^* = \neg(\Sigma^* \cdot (\Sigma - V) \cdot \Sigma^*),$$

---

[1]) Через $P(M)$ обозначается множество всех подмножеств множества M. — *Прим. перев.*

[2]) Если α и β являются γ-выражениями над Σ. — *Прим. перев.*

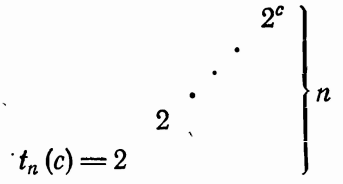
так как V^* состоит в точности из тех слов в Σ^*, которые не содержат символа, не принадлежащего V. Таким образом, существует γ-выражение α над Σ, такое, что $L(\alpha) = V^*$. Законы де Моргана дают нам пересечение, а тогда из тождеств

$$V^n = \Sigma^n \cap V^* \quad \text{и} \quad \Sigma^n = \gamma(V^n)$$

следует, что из γ-выражения длины s для Σ^n мы можем получить γ-выражение длины $s + c$ для V^n при некоторой константе c и всех s, $n \in \mathbb{N}$ и обратно, от V^n к Σ^n. Мы покажем ниже, что вообще s может быть много меньше, чем n.

Определение. $\mathrm{Empty}(\Sigma) = \{\alpha / \alpha$ есть γ-выражение над Σ и $L(\alpha) = \varnothing\}$.

Так как регулярные (распознаваемые конечными автоматами) подмножества Σ^* замкнуты относительно $\cdot$, U, $\neg$ и γ, то отсюда следует, что $\mathrm{Empty}(\Sigma)$ рекурсивно и в действительности примитивно рекурсивно. Легко построить конечный автомат для $L(\alpha)$ и проверить, допускает ли автомат некоторое слово; для этого существуют хорошо известные процедуры. Априорный анализ такой процедуры, однако, показывает, что из детерминированных автоматов для γ-выражений α, β получился бы недетерминированный автомат для $\alpha \cdot \beta$ или $\gamma(\alpha)$, и тогда пришлось бы применить «подмножественную конструкцию» Рабина — Скотта, чтобы получить автомат для $\neg(\alpha \cdot \beta)$ или $\neg\gamma(\alpha)$. Поскольку подмножественная конструкция может экспоненциально увеличить число состояний в автомате, γ-выражения, в которых k дополнений чередуются с γ и конкатенациями, могут привести к автомату с $t_k(2)$ состояниями. Время и зона, необходимые машине Тьюринга, которая распознает $\mathrm{Empty}(\Sigma)$, согласно процедуре, описанной в общих чертах выше, *могут* быть ограничены сверху при некоторой константе c величиной



для всех γ-выражений длины $n \geqslant 0$. Из приводимых ниже результатов следует, что такая абсурдная неэффективность неизбежна.

Определение. Машина Тьюринга $\mathfrak{M}$ распознает множество $A \subset \Sigma^*$, если, будучи запущена с произвольным словом $x \in \Sigma^*$ на ленте, $\mathfrak{M}$ останавливается в указанном допускающем состоянии тогда и только тогда, когда $x \in A$.

Пусть $f\colon \mathbb{N} \to \mathbb{N}^+$ [1]). Емкостная сложность множества $A \subset \Sigma^*$ не превосходит f (не превосходит f почти всюду), записываем

$$\mathrm{Comp}\,(A) \leqslant f \quad (\mathrm{Comp}\,(A) \leqslant f \text{ (п. в.)}),$$

тогда и только тогда, когда существует машина Тьюринга, которая распознает A и для всякого $x \in A$ (для всех $x \in A$, за исключением конечного числа) использует в своем вычислении на входе x не более $f(l(x))$ клеток ленты. *Емкостная сложность A превосходит f бесконечно часто*; записываем

$$\mathrm{Comp}\,(A) > f \text{ (б. ч.)},$$

если не верно, что $\mathrm{Comp}\,(A) \leqslant f$ (п. в.).

Мы используем одну исходную тьюринговскую ленту, модель машины Тьюринга с одной считывающе-записывающей головкой и определяем число клеток ленты, используемых в процессе вычисления на входе x, как максимум из $l(x)$ и числа клеток ленты, посещаемых считывающе-записывающей головкой. Тогда по определению на любом входном слове x в вычислении используется по крайней мере $\max\{l(x), 1\}$ клеток ленты.

Мы кратко повторяем хорошо известные факты о вычислениях на машине Тьюринга с ограниченной зоной, впервые установленные Хартманисом, Стирнзом и Льюисом.

Определение. Функция $f\colon \mathbb{N} \to \mathbb{N}^+$ ленточно конструируема, если $f \geqslant \lambda n[n+1]$ и существует машина Тьюринга, которая, будучи запущена на произвольном входном слове длины $n \geqslant 0$, останавливается, использовав точно $f(n)$ клеток ленты.

Факт 1. $t_0 + 1 = \lambda n[n+1]$ ленточно конструируема. Для всякого $k > 0$ t_k ленточно конструируема.

Факт 2. Если $f\colon \mathbb{N} \to \mathbb{N}^+$ ленточно конструируема и для некоторого $A \subset \Sigma^*$, $\mathrm{Comp}\,(A) \leqslant f$ (п. в.), то $\mathrm{Comp}\,(A) \leqslant f$. В действительности существует машина Тьюринга, которая распознает A и которая останавливается при всяком входе $x \in \Sigma^*$, используя не более $f(l(x))$ клеток ленты. Следовательно, $\mathrm{Comp}\,(A) \leqslant f \Leftrightarrow \mathrm{Comp}\,(\Sigma^* - A) \leqslant f$.

Факт 3. Если $f\colon \mathbb{N} \to \mathbb{N}^+$ ленточно конструируема, то существует $A \subset \{0, 1\}^*$, такое, что $\mathrm{Comp}\,(A) \leqslant f$ и $\mathrm{Comp}\,(A) > g$ (б. ч.) для любой $g\colon \mathbb{N} \to \mathbb{N}^+$, такой, что

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 0.$$

Наше доказательство состоит из последовательности сводимостей одной проблемы разрешения или распознавания к дру-

[1]) Здесь $\mathbb{N} = \{1, 2, 3, \ldots\}$, $\mathbb{N}^+ = \{0, 1, 2, \ldots\}$. — *Прим. перев.*

гой. В противоположность обычным сводимостям теории рекурсивных функций наши сводимости должны быть вычислительно эффективными. Мы вводим особое понятие эффективной сводимости, которое достаточно для наших целей.

Определение. Пусть Σ_1, Σ_2 — конечные множества символов и $A_1 \subset \Sigma_1^*$, $A_2 \subset \Sigma_2^*$. A_1 эффективно сводимо к A_2, записываем A_1 eff A_2, если существует полином p и машина Тьюринга, которая, будучи запущена с произвольным словом $x \in \Sigma_1^*$ на ленте, останавливается со словом $y \in \Sigma_2^*$ на ленте, таким, что

1) $x \in A_1 \Leftrightarrow y \in A_2$ и

2) число клеток ленты, используемых в вычислении для входа x, не превосходит $p(l(x))$ (и тем более $l(y) \leqslant p(l(x))$).

Заметим, что все сводимости, которые описаны ниже, требуют лишь линейного полинома по зоне и только по времени произвольного полинома. Однако, чтобы свести к минимуму требования к интуиции читателей (поскольку мы фактически никогда не приводим диаграмму переходов или таблицу четверок для машин Тьюринга, которые описываем), мы допускаем полиномы любой степени. Точно так же eff ограничено гораздо больше, чем это необходимо, чтобы доказать теорему 2.

Факт 4. eff является транзитивным отношением на множествах слов.

Факт 5. Если A_1 eff A_2 и $\mathrm{Comp}(A_2) \leqslant f$ (п. в.), то существует полином p, такой, что

$$\mathrm{Comp}(A_1) \leqslant \lambda n\,[\max\{f(m) \mid m \leqslant p(n)\} + p(n)] \text{ (п. в.).}$$

Факт 6. Если A_1 eff A_2 и $\mathrm{Comp}(A_1) > t_{k+1}$ (б. ч.), то $\mathrm{Comp}(A_2) > t_k$ (б. ч.).

Доказательство. Непосредственно следует из факта 5 и замечания, что для любого полинома p $t_k(p(n)) + p(n) \leqslant t_{k+1}(n)$ для всех достаточно больших n.

Теперь можно получить доказательство теоремы 2.

Доказательство теоремы 2. Мы установим ниже, что Empty($\{0, 1\}$) eff *WS1S*, Empty(Σ) eff Empty ($\{0, 1\}$) для любого конечного Σ и, наконец, что для любого k и любого множества $A \subset \{0, 1\}^*$, такого, что $\mathrm{Comp}(A) \leqslant t_k$, существует конечное Σ, такое, что A eff Empty (Σ).

Мы имеем, согласно факту 4, A eff *WS1S* для любых A и k, таких, что $\mathrm{Comp}(A) \leqslant t_k$.

Тогда из фактов 1, 3 и 6 заключаем, что для любого k $\mathrm{Comp}(WS1S) > t_{k-1}$ (б. ч.).

Остается лишь установить требуемые сводимости.

Лемма 1. Empty ($\{0, 1\}$) eff *WS1S*.

Доказательство. Мы покажем, как для всякого γ-выражения α над $\{0, 1\}$ построить формулу $F_\alpha \in L_{S1S}$ с двумя свободными числовыми переменными и одной свободной множественной переменной. Для любого множества $M \subset \mathbb{N}$ пусть C_M: $\mathbb{N} \to \{0, 1\}$ есть характеристическая функция M ($C_M(n) = 1 \Leftrightarrow n \in M$). Формула F_α будет построена так, что для $n, m \in \mathbb{N}$, конечного $M \subset \mathbb{N}$

$$F_\alpha(n, m, M) \text{ истинно} \Leftrightarrow [[n < m \text{ и } C_M(n)^{\cdot} C_M(n+1) \ldots$$
$$\ldots C_M(m-1) \in L(\alpha)] \text{ или } [n = m \text{ и } \lambda \in L(\alpha)]].$$

F_α строится индукцией по определению γ-выражений. Если α есть 0 или 1, то

$$F_0(x, y, X) \text{ есть } [y = x+1 \text{ и } \neg(x \in X)],$$
$$F_1(x, y, X) \text{ есть } [y = x+1 \text{ и } x \in X].$$

Если α есть $(\beta^{\cdot}\delta)$, то

$$F_\alpha(x, y, X) \text{ есть } (\exists z)[x \leqslant z \text{ и } z \leqslant y \text{ и } F_\beta(x, z, X) \text{ и } F_\delta(z, y, X)].$$

Если α есть $\gamma(\beta)$, то

$$F_\alpha(x, y, X) \text{ есть } \exists X_0 [F_\beta(x, y, X_0)].$$

Если α есть $(\beta \cup \delta)$ или $\neg(\beta)$, то F_α есть $[F_\beta$ или $F_\delta]$ или $[\neg F_\beta(x, y, X)$ и $x \leqslant y]$ соответственно.

Ясно, что существует машина Тьюринга, которая по данному входу $\alpha \in \{0,1, (,) \cup, \gamma, ^{\cdot}, \neg\}^*$ может проверить, является ли α правильно построенным γ-выражением, и если это так, печатает в качестве выхода предложение

$$\neg(\exists x)(\exists y)(\exists X)[F_\alpha(x, y, X)],$$

используя зону, не превосходящую некоторого фиксированного полинома от $l(\alpha)$. (Если α не является правильно построенным, машина печатает в качестве выхода $(\exists x)[x = x+1]$.) Следовательно, Empty $(\{0, 1\})$ eff $WS1S$.

Нам будет удобнее работать с более широкими множествами символов, нежели $\{0, 1\}$, однако тривиальное кодирование покажет, что это не влечет потери общности.

Пусть Σ — любое конечное множество символов, $|\Sigma| \geqslant 2$, скажем, $0, 1 \in \Sigma$, $0 \neq 1$. Тогда для любого $n \geqslant 1$ существует γ-выражение над Σ для $(\Sigma^n)^*$. Чтобы убедиться в этом, рассмотрим слово из Σ^*, *не* принадлежащее $(0^{n-1}1)^*$. Такое слово либо не начинается с $0^{n-1}1$, либо не оканчивается на 1, либо содержит подслово из $0 \cdot \Sigma^{n-1}(\Sigma - 0)$ или $1 \cdot \Sigma^{n-1}(\Sigma - 1)$. Следовательно, $\neg((0^{n-1}1)^*) \cup \lambda = \neg(0^{n-1}1\Sigma^*) \cup \neg(\Sigma^* \cdot 1) \cup (\Sigma^* 0 \Sigma^{n-1}(\Sigma - 0)\Sigma^*) \cup$

$\cup(\Sigma^* 1 \Sigma^{n-1}(\Sigma - 1)\Sigma^*)$, и, замечая, что $\lambda = \neg(\Sigma \cdot \Sigma^*)$, мы имеем $(0^{n-1}1)^* = \neg(\lambda \cup \neg((0^{n-1}1)^*)) \cup \lambda$ и $(\Sigma^n)^* = \gamma((0^{n-1}1)^*)$.

Если теперь дано любое конечное множество Σ_1, выберем n достаточно большим, с тем чтобы $|\Sigma^n| \geqslant |\Sigma_1|$, и пусть h: $\Sigma_1 \to \Sigma^n$ — любая однозначная функция. Распространим h до однозначного отображения $P(\Sigma_1^*)$ в $P((\Sigma^n)^*)$ с помощью очевидных правил: $h(\lambda) = \lambda$, $h(x\sigma_1) = h(x)^* h(\sigma_1)$ для $x \in \Sigma_1^*$, $\sigma_1 \in \Sigma_1$, и $h(A) = \bigcup\limits_{x \in A} \{h(x)\}$ для $A \subset \Sigma_1^*$. Тогда существует γ-выражение над Σ для $h(\Sigma_1^*)$, поскольку слово не принадлежит $h(\Sigma_1^*)$ либо потому, что его длина не кратна n, либо еще потому, что оно содержит подслово длины n, не принадлежащее $h(\Sigma_1)$, которое начинается с позиции, сравнимой с первой по модулю n:

$$\Sigma^* - h(\Sigma_1^*) = \neg((\Sigma^n)^* \cup (\Sigma^n)^* \cdot (\Sigma^n - h(\Sigma_1)) \cdot (\Sigma^n)^*).$$

Лемма 2. (Кодирование.) *Пусть* Σ_1, Σ — *конечные множества символов, причем* $|\Sigma| \geqslant 2$. *Пусть для некоторого* $n \geqslant 1$ h: $P(\Sigma_1^*) \to P((\Sigma^n)^*)$ — *распространение однозначной функции из* Σ_1 *в* Σ^n. *Тогда существует машина Тьюринга, которая, будучи запущена с* γ-*выражением* α *над* Σ_1, *останавливается с* γ-*выражением* β *над* Σ *на ленте, таким, что*

$$h(L(\alpha)) = L(\beta).$$

Кроме того, зона, используемая в процессе вычисления с входом α, *ограничена полиномом от* $l(\alpha)$.

Доказательство. Преобразование α в β происходит путем рекурсивного применения следующих правил:

Если $\alpha \in \Sigma_1$, то β есть выражение, равное некоторому выражению для $h(L(\alpha))$.

Если α есть $(\alpha_1^* \alpha_2)$ или $(\alpha_1 \cup \alpha_2)$, то β есть $(\beta_1^* \beta_2)$ или $(\beta_1 \cup \beta_2)$ соответственно, где β_1, β_2 — преобразования α_1, α_2. Если α есть $\gamma(\alpha_1)$, то β есть

$$\neg((\neg(\gamma(\beta_1)) \cup \beta_{\Sigma_1})),$$

где β_1 — преобразование α_1 и β_{Σ_1} — γ-выражение над Σ для $\Sigma^* - h(\Sigma_1^*)$. (Заметим, что для $A \subset \Sigma_1^*$ $h(\gamma_{\Sigma_1}(A)) = \gamma_\Sigma(h(A)) \cap h(\Sigma_1^*)$, что является обоснованием этого правила.) Наконец, если α есть $\neg(\alpha_1)$, то β есть $\neg((\beta_1 \cup \beta_{\Sigma_1}))$, так как для $A \subset \Sigma_1^*$ $h(\Sigma_1^* - A) = h(\Sigma_1^*) - h(A) = \Sigma^* - (h(A) \cup (\Sigma^* - h(\Sigma_1^*)))$.

Ясно, что машина Тьюринга может выполнить это рекурсивное преобразование в пределах требуемой границы зоны.

Следствие. *Для любого конечного* Σ

$$\text{Empty}(\Sigma) \text{ eff } \text{Empty}\{0, 1\}.$$

Доказательство. Закодируем Σ в $\{0, 1\}$ с помощью h как в лемме 2. Тогда $\alpha \in \mathrm{Empty}(\Sigma) \Leftrightarrow L(\alpha) = \varnothing \Leftrightarrow h(L(\alpha)) = \varnothing \Leftrightarrow L(\beta) = \varnothing \Leftrightarrow \beta \in \mathrm{Empty}\{0, 1\}$. ч. т. д.

Пусть дано γ-выражение для Σ^n. Мы теперь показываем, как можно построить γ-выражение приблизительно такого же размера, описывающее любое требуемое вычисление машины Тьюринга при условии, что состояния и символы машины Тьюринга можно представить в Σ и что вычисление требует только n клеток ленты. Это построение будет выполняться рекуррентно, с тем чтобы получить γ-выражения размера n для $\Sigma^{t_k(n)}$, и потом, в конце, будет использовано для того, чтобы сделать вывод, что A eff Empty (Σ) для любого $A \subset \{0, 1\}^*$, удовлетворяющего соотношению $\mathrm{Comp}(A) \lesssim t_k$.

Определение. Пусть $\mathfrak{M}$ — произвольная машина Тьюринга с множеством ленточных символов T и множеством состояний S. Предполагаем, что $b \in T$, где b обозначает пустую клетку ленты. *Мгновенное описание* (м. о.) машины $\mathfrak{M}$ есть слово из $(T \cup (S \times T))^*$, которое содержит точно один символ из $S \times T$ [1]). Если дано произвольное м. о. $x = y^\cdot(s, t)^\cdot z$ для $y, s \in T^*$, следующее м. о., $\mathrm{Next}_{\mathfrak{M}}(x)$, определяется следующим образом: если $\mathfrak{M}$, находясь в состоянии s и обозревая своей считывающе-записывающей головкой символ t, переходит в состояние s' и печатает символ $t' \in T$, то $\mathrm{Next}_{\mathfrak{M}}(x)$ есть

$y^\cdot(s', t')^\cdot z$, если $\mathfrak{M}$ не сдвигает свою головку,

$y^\cdot t'^\cdot(s', u)^\cdot w$, если $\mathfrak{M}$ сдвигает свою головку вправо и $z = uw$ для $u \in T$, $w \in T^*$,

$w^\cdot(s', u)^\cdot t'^\cdot z$, если $\mathfrak{M}$ сдвигает свою головку влево и $y = wu$ для $u \in T$, $w \in T^*$.

$\mathrm{Next}_{\mathfrak{M}}(x)$ не определено, если (s, t) есть условие останова или если (s, t) есть правый (левый) крайний символ x и $\mathfrak{M}$ сдвигается вправо (влево). Пусть $\mathrm{Next}_{\mathfrak{M}}(x, 0) = x$, если x является м. о., и не определено в противном случае; $\mathrm{Next}_{\mathfrak{M}}(x, n+1) = \mathrm{Next}_{\mathfrak{M}}(\mathrm{Next}_{\mathfrak{M}}(x, n))$.

Наконец, пусть $\#$ — символ, не принадлежащий $T \cup (S \times T)$. Вычисление $c_{\mathfrak{M}}$ машины $\mathfrak{M}$ на x есть следующее слово в $(\{\#\} \cup T \cup \{S \times T\})^*$:

$$c_{\mathfrak{M}}(x) = \#^\cdot \mathrm{Next}_{\mathfrak{M}}(x, 0)^\cdot \#^\cdot \mathrm{Next}_{\mathfrak{M}}(x, 1)^\cdot \#^\cdot \ldots {}^\cdot \# \mathrm{Next}_{\mathfrak{M}}(x, n)^\cdot \#,$$

где n — наименьшее целое, такое, что (q_a, t) встречается в $\mathrm{Next}(x, n)$ для некоторого $t \in T$ и указанного заключительного состояния q_a. Вычисление $c_{\mathfrak{M}}(x)$ не определено, если такого n не существует.

---

[1]) $S \times T = \{(s, t) \mid s \in S \text{ и } t \in T\}$.

Замечание. Заметим, что наше определение вычисления отличается от определения, принятого в литературе. Вычисление $c_{\mathfrak{M}}(x)$ определяется для мгновенных описаний x, а *не* для входных слов x. Более того, все м. о. в $c_{\mathfrak{M}}(x)$ имеют в точности одну и ту же длину. Ниже приводится ключевое свойство $c_{\mathfrak{M}}(x)$.

Факт 7. Пусть $\mathfrak{M}$ задана как в предыдущем определении, и пусть $\Sigma = \{\#\} \cup T \cup \{S \times T\}$. Для любого м. о. x пусть $c_{\mathfrak{M}}(i, x)$ — i-символ в $c_{\mathfrak{M}}(x)$, $1 \leqslant i \leqslant l(c_{\mathfrak{M}}(x))$. Тогда существует частичная функция $f_{\mathfrak{M}}$: $\Sigma^3 \to \Sigma$, такая, что для любого м. о. x и любого целого i, $l(x) + 2 < i \leqslant l(c_{\mathfrak{M}}(x))$,

$$c_{\mathfrak{M}}(i, x) = f_{\mathfrak{M}}(c_{\mathfrak{M}}(i - (l(x) + 2), x), c_{\mathfrak{M}}(i - (l(x) + 1), x),$$
$$c_{\mathfrak{M}}(i - l(x); x) \quad \text{и} \quad f_{\mathfrak{M}}(\sigma_1, \sigma_2, \sigma_3) = \varnothing,$$

если $\sigma_i \in S \times T$ является условием останова $\mathfrak{M}$, $i = 1, 2, 3$. Это непосредственно следует из того, что $(n-1)$-й, n-й и $(n+1)$-й символы любого м. о. y единственным образом определяют n-й символ $\mathrm{Next}_{\mathfrak{M}}(y)$ при условии, что $\mathrm{Next}_{\mathfrak{M}}(y)$ определено.

Лемма 3. (Моделирование.) *Пусть $\mathfrak{M}$ — машина Тьюринга с множеством состояний S, множеством символов T и указанным заключительным состоянием $q_a \in S$. Пусть $\Sigma = \{\#\} \cup T \cup (S \times T)$. Существует машина Тьюринга $C_{\mathfrak{M}}$, которая, будучи запущена с произвольным словом $y^* \#^* \alpha$ на ленте, где y — м. о. машины $\mathfrak{M}$ и α — γ-выражение над Σ, такое, что $L(\alpha) = \Sigma^n$ для некоторого $n > 0$, останавливается с γ-выражением β над Σ, таким, что*

$$L(\beta) = \begin{cases} \{c_{\mathfrak{M}}(b^n y b^n)\}, & \text{если } c_{\mathfrak{M}}(b^n y b^n) \text{ определено,} \\ \varnothing & \text{в противном случае.} \end{cases}$$

Кроме того, существует полином p, такой, что $C_{\mathfrak{M}}$ использует в своем вычислении не более $p(l(y^* \#^* \alpha))$ клеток ленты.

Доказательство. Мы опишем, как построить из $y^* \#^* \alpha$, где $L(\alpha) = \Sigma^n$, γ-выражение β для $\{c_{\mathfrak{M}}(b^n y b^n)\}$. Мы начинаем с замечания о том, что слова из Σ^*, *не* равные $c_{\mathfrak{M}}(b^n y b^n)$, т. е. $\neg(c_{\mathfrak{M}}(b^n y b^n))$, можно охарактеризовать следующим образом:

1) слова, которые не начинаются с $\# b^n y b^n \#$, или
2) слова, которые не содержат q_a, или
3) слова, которые не оканчиваются на $\#$, или
4) слова, которые не удовлетворяют функциональному условию, определенному функцией $f_{\mathfrak{M}}$ в факте 7.

Эти четыре множества слов можно также описать с помощью формул

1′) $\neg(\#^* (L(\alpha) \cap b^*)^* y^* (L(\alpha) \cap b^*)^* \#^* \Sigma^*)$,
2′) $\neg(\Sigma^{**} (\{q_a\} \times T)^* \Sigma^*)$,

3′) $\neg(\Sigma^{*}\cdot\#)$,

4′) $\bigcup_{\sigma_1,\sigma_2,\sigma_3\in\Sigma}\left[\Sigma^{*}\cdot\sigma_1\cdot\sigma_2\cdot\sigma_3 L(\alpha)\cdot\Sigma^{l(y)-1}L(\alpha)\cdot(\Sigma-f_{\mathfrak{M}}(\sigma_1,\sigma_2,\sigma_3))\cdot\Sigma^{*}\right]$.

Легко, однако, видеть, как построить γ-выражение прямо из (1′)—(4′), и, следовательно, β просто является дополнением к объединению этих четырех выражений. Заметим, что $l(\beta)\leqslant c\cdot l(y\#\alpha)$ для некоторой константы c, которая зависит только от $\mathfrak{M}$, но не от y или α. Кроме того, машине Тьюринга $C_{\mathfrak{M}}$, которая строит β из $y\#\alpha$, нет необходимости использовать больше чем $l(\beta)$ клеток ленты, и она, таким образом, конечно, двигается в пределах полиномиальной границы зоны.

Определение. $\Sigma-t_k-MT$ есть машина Тьюринга, такая, что для некоторого полинома p, некоторой функции $f_k\geqslant t_k$ и всех $n>0$, если машина запущена с 0^n на ленте, она останавливается со словом α на ленте, таким, что

1) α является γ-выражением над Σ и $L(\alpha)=\Sigma^{f_k(n)}$,

2) число клеток ленты, используемых в вычислении, не превосходит $p(n)$.

Лемма 4. *Если для какого-нибудь конечного* Σ' *существует* $\Sigma'-t_k-MT$, *то и для любого* Σ, *такого, что* $|\Sigma|\geqslant 2$, *существует* $\Sigma-t_k-MT$.

Доказательство. Как и в лемме 2, кодируем Σ' в Σ. Детали оставляются читателю.

Лемма 5. *Для любого* $k\geqslant 0$ *и любого* Σ, $|\Sigma|\geqslant 2$, *существует* $\Sigma-t_k-MT$.

Доказательство. $\Sigma-t_0-MT$ просто печатает по входу 0^n некоторое выражение для $\gamma(\sigma^n)$, где $\sigma\in\Sigma$ — любой символ. Продолжая по индукции, предположим, что существует $\Sigma-t_k-MT$. Пусть $\mathfrak{M}_k$ — машина Тьюринга, которая, будучи запущена с 0^n на ленте при произвольном $n>0$, отмечает $t_k(n)$ клеток на ленте и затем использует эти клетки, чтобы циклически работать в течение некоторого числа $f_{k+1}(n)\geqslant 2^{t_k(n)}=t_{k+1}(n)$ шагов вплоть до окончательного останова. Выберем Σ как в лемме о моделировании, примененной к $\mathfrak{M}_k$. $\Sigma-t_{k+1}-MT$ работает следующим образом: если дано 0^n, используем $\Sigma-t_k-MT$, чтобы получить α, такое, что $L(\alpha)=\Sigma^{t_k(n)}$. Применяем $C_{\mathfrak{M}_k}$ из леммы о моделировании к $(q_0,0)\,0^{n-1}\cdot\#\cdot\alpha$, где q_0 — начальное состояние $\mathfrak{M}_k$. Это дает γ-выражение β, такое, что $L(\beta)=\{c_{\mathfrak{M}_k}(x)\}$, где $x=b^{f_k(n)}(q_0,0)\,0^{n-1}b^{f_k(n)}$. Однако $c_{\mathfrak{M}_k}(x)$ определено, поскольку $\mathfrak{M}_k$ при входе 0^n останавливается в пределах $t_k(n)\leqslant f_k(n)$ клеток ленты. Кроме того, $l(c_{\mathfrak{M}_k}(x))\geqslant t_{k+1}(n)$, так как $\mathfrak{M}_k$ делает

по крайней мере $t_{k+1}(n)$ шагов. Следовательно, выход Σ—t_{k+1}—MT есть просто $\gamma(\beta)$. Так как α получается на зоне $p_1(n)$ для некоторого полинома p_1 и β подобным образом получается на зоне $p_2(n+1+p_1(n))$ для некоторого полинома p_2, то весь процесс требует полиномиальной зоны.

Лемма 6. *Для любого множества* $A \subset \{0, 1\}^*$, *если для некоторого* $k \geqslant 0$ $\mathrm{Comp}(A) \leqslant t_k$, *то существует конечное* Σ, *такое, что* A eff $\mathrm{Empty}(\Sigma)$.

Доказательство. Пусть $\mathfrak{M}$ — машина Тьюринга, которая распознает $\{0, 1\}^* - A$ и для любого $x \in \{0, 1\}^*$ $\mathfrak{M}$ останавливается, использовав не более $t_k(l(x))$ клеток ленты. Такая $\mathfrak{M}$ существует согласно факту 2.

Выберем Σ как в лемме о моделировании, примененной к $\mathfrak{M}$. Машина Тьюринга, которая эффективно сводит A к $\mathrm{Empty}(\Sigma)$, работает следующим образом: если дано $x \in \{0, 1\}^*$, используем Σ—t_k—MT, чтобы получить γ-выражение α, такое, что $L(\alpha) = \Sigma^{t_k(n)}$ для $n = l(x)$. Применяем $C_{\mathfrak{M}}$ из леммы о моделировании к $(q_0, u)^* w^* \#^* \alpha$, где q_0 — начальное состояние $\mathfrak{M}$ и $x = uw$ для $u \in \{0, 1\}$, $w \in \{0, 1\}^*$. (Мы опускаем случай $x = \lambda$.) Это дает γ-выражение β, которое, как мы утверждаем, является искомым выходом.

Так как машине $\mathfrak{M}$ необходима зона не более $t_k(n)$, то вычисление $c_{\mathfrak{M}}(y)$, где $y = b^{t_k(n)}(q_0, u)^* w^* b^{t_k(n)}$, определено тогда и только тогда, когда x допускается машиной $\mathfrak{M}$. Следовательно, $x \in A \Leftrightarrow x$ не допускается машиной $\mathfrak{M} \Leftrightarrow c_{\mathfrak{M}}(y)$, не определено $\Leftrightarrow L(\beta) = \varnothing \Leftrightarrow \beta \in \mathrm{Empty}(\Sigma)$. Это обосновывает наше утверждение о том, что β является правильным выходом.

Как и в предыдущей лемме, машине Тьюринга, преобразующей x в β, необходима зона, не превосходящая некоторого полинома от $l(x)$.

Эта лемма завершает леммы, необходимые для теоремы 2.

Нетрудно обобщить это доказательство так, чтобы получить теорему 1. Мы пользуемся более строгой формой факта 3 для того, чтобы сделать вывод из доказательства теоремы 2 об информации относительно частоты повторения (м. о.)-условия в утверждении о том, что $\mathrm{Comp}(WS1S) > t_k$ (б. ч.).

Следствие. *Следующие разрешимые полные и слабые теории второго порядка не элементарно рекурсивны: двух функций следования, счетного линейного порядка, функции одного аргумента со счетной областью определения, единичного интервала относительно* $\leqslant$. (*Разрешимая*) *теория первого порядка двух функций следования с предикатами длины и префикса также не элементарно рекурсивна. Теория первого порядка сложения и Р*

на неотрицательных целых числах, где $P(x, y) \equiv$ [*x есть степень двойки, которая делит y*], *разрешима, но не элементарно.*

Это вытекает из подходящих прямых эффективных редукций $WS1S$ к каждой из указанных теорий.

γ-выражения сами по себе представляют интерес в качестве разрешимой, но не элементарной словарной проблемы.

Следствие. Empty({0, 1}) *не элементарно рекурсивно.*

Дальнейшие замечания: (1) Рабин утверждает в своей работе о разрешимости $S2S$-(полной) сингулярной теории второго порядка двух функций следования, что его процедура элементарно рекурсивна. По-видимому, он был введен в заблуждение тем фактом, что каждый шаг элиминации кванторов в его доказательстве элементарен, и тем, что заключительная проверка, допускает ли автомат на дереве непустое множество, также является элементарной относительно размера автомата. Тем не менее последовательная элиминация кванторов составляет элементарные шаги и дает процедуру, требующую зону приблизительно $t_n(2)$ для формул длины n.

(2) У нас еще не было времени, чтобы рассмотреть, для каких разрешимых теорий, кроме тех, которые отмечены в заключительных следствиях, можно этими методами показать неэлементарность. Направления дальнейшего исследования совершенно очевидны.

(3) Построение в лемме о моделировании можно выполнить с использованием обычных регулярных выражений (с $\cdot$, $\cup$ и $*$ в качестве единственных операций), хотя с несколько большим усилием. Стандартные методы теории автоматов показывают, что общая проблема эквивалентности регулярных выражений имеет верхнюю границу по зоне n^2 и по времени d^n для некоторого $d > 1$.

Подобным образом, если рассматривать регулярные выражения с сокращением α^2 для $\alpha{\cdot}\alpha$, то любая процедура для решения, равны ли регулярные выражения с возведением в квадрат выражению $\{0, 1\}^*$, требует *зоны* c^n для некоторого $c > 1$ [1]). Общая проблема эквивалентности вновь может быть решена с зоной d^n и временем d^{d^n} для некоторого $d > 1$. Эти результаты появятся в совместной с Л. Стокмейером статье, более ранняя работа которого по регулярным выражениям подсказала лемму о моделировании.

Таким образом, наша техника не ограничена процедурами, для которых элементарность не известна. Надеемся, что мы

[1]) Речь идет, конечно, о равенстве множеств, задаваемых регулярными выражениями. — *Прим. перев.*

в состоянии получать нетривиальные нижние границы сложности разрешения проблем, для которых элементарность известна.

(4) Некоторые занимательные технические вопросы о γ-выражениях к настоящему моменту остаются открытыми. Мы можем показать, что не все регулярные множества представимы в виде γ-выражений, однако наше доказательство привлекает некоторые глубокие результаты алгебраической теории автоматов. Наше доказательство не исключает возможности того, что множество слов в $\{0, 1\}^*$ с четным числом нулей и произвольным числом единиц γ-выразимо, однако мы не смогли найти γ-выражение для этого множества. Заметим, что $(00)^*$ — γ-выразимо (ср. с леммой 2).

(5) Абстрактная теория сложности оставалась открытой для критики из-за невозможности указать «естественные» проблемы разрешения, в которых появлялись такие явления, как ускорение. Применяя результаты Блюма об эффективном ускорении к нашему моделированию машин Тьюринга посредством *WS1S*, мы можем показать, что если дана произвольная разрешающая процедура для *WS1S*, то можно эффективно построить новую разрешающую процедуру для *WS1S*, которая много быстрее (ускорение с помощью t_k при любом k), чем данная процедура, по крайней мере на одном предложении длины n для всех достаточно больших целых n.

(6) Отношение eff можно охарактеризовать путем, подобным определению элементарных функций или примитивно рекурсивных функций. Класс $\mathscr{E}^{2,5}$, названный так потому, что он расположен строго между классами Гжегорчика $\mathscr{E}^2$ и $\mathscr{E}^3$, определяется по индукции следующим образом:

1. $x \dot{-} y,\ x + y,\ xy,\ x^{[\log_2 y]} \in \mathscr{E}^{2,5}$.
2. $\mathscr{E}^{2,5}$ замкнут относительно явных преобразований (подстановки констант и переименования или отождествления переменных),
3. $\mathscr{E}^{2,5}$ замкнут относительно суперпозиции функций и
4. $\mathscr{E}^{2,5}$ замкнут относительно ограниченной рекурсии, ограниченного суммирования и ограниченной минимизации [1]).

Предположим, что мы отождествляем слова из Σ^* с целыми числами, которые они представляют в системе счисления с основанием Σ, и пусть для любого множества $A \subset \Sigma^*$ характеристическая функция множества целых чисел, отождествляемого с A, есть C_A: $\mathbb{N} \to \{0, 1\}$. Тогда B eff A тогда и только тогда, когда $C_B(x) = C_A(f(x))$ для некоторой $f \in \mathscr{E}^{2,5}$ и всех $x \in \mathbb{N}$.

---

[1]) Определения смотри в статье Гжегорчика. Замкнутость относительно ограниченной рекурсии на самом деле влечет замкнутость относительно ограниченного суммирования и ограниченной минимизации.

По существу, $\mathscr{E}^{2,5}$ доставляет язык программирования высокого уровня, в котором можно формально выразить процедуры, о которых мы неформально утверждали, что они могут быть выполнены на машинах Тьюринга с полиномиально ограниченной зоной. Этим способом наше доказательство можно представить в совершенно формальной форме без обращения к интуитивным соображениям относительно потребностей зоны при вычислениях. Мы предпочитаем последний подход.

Доказательство Ларри Стокмейером того, что любая проблема, недетерминированно разрешимая за полиномиальное время, детерминированно сводима за полиномиальное время к регулярному выражению, не равному Σ^*, дало ключевую идею к лемме о моделировании. Джин Ферранте и Чарльз Раков выполнили эффективные сведения $WS1S$ к другим упомянутым теориям. Патрик Фишер корректно указал на то, что использование $*$ в моем первоначальном доказательстве было несущественным.

Советы и внимание моего коллеги Майкла Дж. Фишера были в высшей степени полезными, как они неизменно бывали и в прошлом.

ПРИМЕЧАНИЯ ПЕРЕВОДЧИКА

Спустя полтора года после выхода в свет настоящей работы появилась новая работа А. Мейера под тем же заголовком. В ней с небольшими техническими изменениями изложены те же результаты, что и в настоящей работе. Сообщается также, что автором совместно с М. Фишером, М. Рабином, Л. Стокмейером, Дж. Ферранте и Ч. Раковом получены близкие верхние и нижние оценки зоны и времени разрешающих процедур для многих разрешимых теорий в логике и теории автоматов. Ниже мы приводим некоторые из этих результатов.

1. (А. Мейер.) Проблема выполнимости в теории первой ступени линейного порядка не элементарна. Для подходящих констант c верхняя и нижняя оценки зоны имеют вид $t_{c \cdot n}(n)$.

2. (Л. Стокмейер.) Проблема пустоты для выражений, содержащих только операции $\cup$, $\cdot$, $\neg$, не элементарна.

3. (М. Фишер — М. Рабин.) Любая разрешающая процедура для теории первого порядка $\langle N, +\rangle$, т. е. для арифметики Пресбургера, требует $t_2(c \cdot n)$ шагов даже на недетерминированных машинах Тьюринга.

Верхнюю границу зоны, равную $t_2(c' \cdot n)$, установили Дж. Ферранте и Ч. Раков.

4. (М. Фишер — М. Рабин.) Любая разрешающая процедура для теории первого порядка $\langle N - \{0\}, \cdot\rangle$ требует времени $t_3(c \cdot n)$ даже на недетерминированных машинах Тьюринга.

5. (М. Фишер.) Пусть S — класс структур с бинарным ассоциативным оператором $*$, обладающий тем свойством, что для бесконечно многих n существует $s \in S \in S$, такое, что

$$s^n \neq s^m \quad \text{при} \quad 1 \leqslant m \leqslant n,$$

где $s^m = \underbrace{s * s * \ldots * s}_{m}$. Тогда любая процедура проверки выполнимости в S предложений языка первого порядка сигнатуры $\{*\}$ требует $t_1(c \cdot n)$ шагов. Этот результат применим почти ко всем известным разрешимым теориям в логике, за исключением пропозиционального исчисления и исчисления чистого равенства.

6. (А. Мейер.) Процедура проверки выполнимости предложений примерно с семью кванторами в сингулярном исчислении предикатов требует времени $t_1(c \cdot n)$ даже на недетерминированных машинах Тьюринга. Время $t_1(c' \cdot n)$ достижимо на недетерминированных машинах Тьюринга.

7. (М. Фишер — А. Мейер.) Проблема выполнимости в теории первого порядка единственной одноместной функции не элементарна.

ЛИТЕРАТУРА

1. Blum M., On effective procedures for speed-up algorithms, *Jour. Assoc. Comp. Mach.*, **18**, 2, April 1971, 290—305.
2. Buchi J. R., Weak second-order arithmetic and finite automata, *Zeit. Math. Logik und Grund. Math.*, **6** (1960), 66—92. (Русский перевод: Д. Р. Бюхи, Слабая арифметика второго порядка и конечные автоматы, Кибернетический сборник, 8 (1964), 42—47.)
3. Grzegorczyk A., Some classes of recursive functions, Rozprawy Mathematiczne, 4, Warsaw, 1953, 1—45. (Русский перевод: А. Гжегорчик, Некоторые классы рекурсивных функций, сб. пер. «Проблемы математической логики», М. (1970), 9—49.)
4. Rabin M. O. Decidability of second-order theories and automata on infinite trees, *Trans. Amer. Math. Soc.*, **141** (1969), 1—35.
5. Ritchie R. W., Classes of predictably computable functions *Trans. Amer. Math. Soc.*, **106** (1963), 139—173. (Русский перевод: Р. В. Ричи, Классы предсказуемо вычислимых функций, сб. пер. «Проблемы математической логики», М., 1970, 50—93.)
6. Stearns R. E., Hartmanis J., Lewis P. M. II, Hierarchies of memory-limited computations, IEEE Conf. Rec. on Switching Theory and Logical Design, Sixth Annual Symposium, (1965), 179—190. (Русский перевод: Р. Е. Стирнз, Дж. Хартманис, П. М. Льюис II, Иерархии вычислений с ограниченной памятью, сб. пер. «Проблемы математической логики», М. (1970), 301—319.)

О формализованных машинных программах[1])

Д. Лакхэм, Д. М. Парк, М. С. Патерсон

1. ВВЕДЕНИЕ

В работе [2] Янов ввел простую абстрактную модель машинной программы (под названием схемы программы), основанную на представлении программы в виде конечной линейной последовательности операторов двух типов: вычислительных операторов и операторов условного перехода по одному из двух направлений. Команды перехода, управляющие порядком, в котором выполняются операторы, зависят от значения пропозициональных истинностных функций конечного числа переменных; значения последних (истина или ложь) могут изменяться при выполнении того или иного вычисления. Ни один из вычислительных операторов и ни одна из булевых функций не интерпретированы (т. е. их значения или интерпретация отсутствуют), так что схема программы может быть представлена как семейство машинных программ, каждый член которого является допустимой интерпретацией схемы. Свойства абстрактных схем этого типа не зависят от какой-либо конкретной машины или языка программирования и выполняются в программах, записанных на любом языке, который обладает последовательностными и контролирующими характеристиками, предполагаемыми в модели. Основные результаты, полученные Яновым, заключаются в существовании двух алгоритмов: алгоритма, который для любой пары схем устанавливает, представляют ли они при всех интерпретациях одни и те же программы (т. е. являются эквивалентными) или нет, и алгоритма, переводящего схему в эквивалентную ей простую каноническую форму. Эти результаты кратко излагаются Ратледжем в [3], где, кроме того, отмечается прямая связь между схемами Янова и конечными автоматами (и, таким образом, результаты, полученные в других областях теории вычислений, могут быть проинтерпретированы «программно-ориентированными» средствами).

Одна из основных целей исследования свойств таких абстрактных программ состоит в подходе на некоторой общей основе к проблеме оптимизации программ. Алгоритмы, осущест-

---

[1]) Luckham D. C., Park D. M. R., Paterson M. S., On formalised computer programs, J. Computer and System Sci., 4, 220—249 (1970).

вляющие исключение операторов из схем или слияние и укорачивание циклов в схеме при условии сохранения эквивалентности, должны давать основу для практических упрощений. Однако модель Янова слишком элементарна, и должны быть учтены другие «общие» свойства машинных программ, например зависимость или независимость вычислительных операторов внутри программы.

В разд. 2 мы предлагаем естественное обобщение схемы Янова посредством введения свободных переменных. Вычислительные операторы и операторы перехода задаются нами в виде функций переменных, и значение каждого вычисления присваивается некоторой переменной. Содержательно абстрактная схема теперь изображает «поток информации» в программе.

Хотя наш формализм обладает определенными алголо-подобными особенностями, несущественность этого легко показать; схемы программ могут быть заданы, например, совокупностями рекурсивных уравнений, и поэтому результаты, полученные в последующих разделах, сохраняют свою справедливость для программ, записанных и в этом формализме.

Вопрос разрешимости эквивалентности схем программ тесно связан с существованием упрощающих алгоритмов. Если проблема эквивалентности разрешима, то в принципе существует алгоритм для сведения схемы к простейшей (в некотором смысле) возможной форме. В разделах 4 и 5 мы показываем, что для почти всякого разумного понятия эквивалентности между машинными программами оба вопроса: об эквивалентности и о неэквивалентности пары схем *не являются* частично разрешимыми. (Здесь под «частичной разрешимостью» понимается «рекурсивная перечислимость», а именно: отношение $r(a, b)$ частично разрешимо, но не разрешимо, если существует алгоритм для порождения списка всех пар $\langle a, b\rangle$, таких, что $r(a, b)$ выполняется, но не существует алгоритма, перечисляющего все такие пары, для которых $r(a, b)$ не выполняется.) Отсюда следует, что не существует оптимизирующих алгоритмов для полного сведения схемы k, так сказать, к кратчайшей возможной форме. Фактически, как будет показано, таких алгоритмов не существует даже в случае строго ограниченных классов схем. В разд. 4 устанавливается основной результат о частичной неразрешимости строгой эквивалентности (работа [3] содержит более раннее доказательство), в разд. 5 этот результат распространяется на более слабые понятия эквивалентности. Доказательства используют некоторые свойства многоголовочных автоматов; последние рассматриваются в разд. 3.

В разд. 6 мы устанавливаем эквивалентность между очень простым классом схем и многоленточными автоматами. Отсюда следует, что схемы Янова эквивалентны схемам программ,

содержащим *одну* свободную переменную. Наконец, чтобы нейтрализовать отрицательные результаты, полученные в разд. 4 и 5, мы даем сжатый обзор классов схем, для которых проблема эквивалентности разрешима и существуют оптимизирующие алгоритмы, и перечисляем некоторые открытые проблемы.

2. СХЕМЫ ПРОГРАММ

Введем формальный язык, содержащий следующие символы:

(i) цифры,
(ii) $F_1^1, F_2^1, F_3^1, \ldots, F_1^2, F_2^2, F_3^2, \ldots$ (функциональные символы),
(iii) $L_1, L_2, L_3, \ldots$ (символы ячеек),
(iv) $T_1, T_2, T_3, \ldots$ (предикатные символы),
(v) $:=$ (символ присваивания),
(vi) STOP,
(vii) (,) и запятая (вспомогательные символы).

Допустимыми выражениями (или операторами) языка назовем выражения трех типов (символами, отличными от L, F, STOP, обозначаются числа; в каждом случае k называется префиксом или адресом оператора):

(i) *Операторы присваивания* [1])

$$k.\ L_i := F_j^n(L_{k_1}, L_{k_2}, \ldots, L_{k_n}).$$

(ii) *Операторы перехода*

$$k.\ T_i(L_j)\, m,\ n,$$

где m, n — числа (адреса перехода).

(iii) *Оператор STOP*

$$k.\ \text{STOP}.$$

Схемой программы (обозначим ее символом P) назовем такую конечную последовательность допустимых операторов, что (i) префикс каждого оператора является его позицией в последовательности, (ii) все адреса перехода являются префиксами в P и (iii) последним оператором является или переход, или STOP.

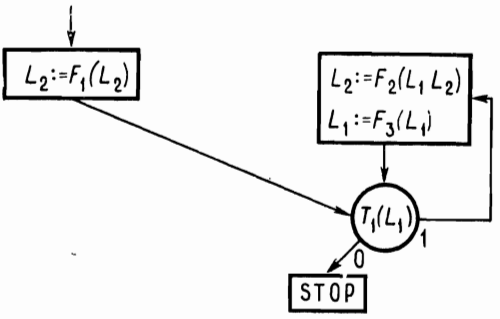
Часто будет удобно изображать схемы в виде потоковых диаграмм; диаграммой 1 изображается схема

$$1.\ L_2 := F_1(L_2),$$
$$2.\ T_1(L_1)\, 5,\ 5,$$
$$3.\ L_2 := F_2(L_1, L_2),$$

[1]) Называемые также вычислительными операторами.

4. $L_1 := F_3(L_1)$,
5. $T_1(L_1)$ 6, 3,
6. STOP.

Заметим, что оператор 2 — это «безусловный» переход, и поэтому он не воспроизводится в потоковой диаграмме в точности.



Содержательно схема программы призвана изображать семейство допустимых машинных программ. Под этим понимается следующее. Как только функциональные и предикатные символы схемы P получают конкретную интерпретацию (посредством функций и характеристических функций соответственно, причем не обязательно конструктивных), схема превращается в программу, которая может быть выполнена идеализированной машиной. Процесс выполнения состоит в последовательном выполнении операторов в том порядке, в котором они следуют друг за другом в схеме (за исключением случаев, когда встречаются операторы перехода), начиная с первого оператора и при начальной совокупности данных, хранящихся в ячейках схемы P; пусть это будут $L_1, L_2, \ldots, L_n$. На каждом шаге выполняется оператор одного из двух типов: если выполняется $L_\omega := F_j(L_{u_1}, \ldots, L_{u_r})$, то результат вполне определенного вычисления, осуществленного над содержимым ячеек $L_{u_1}, \ldots, L_{u_r}$ (которое остается неизменным), присваивается ячейке L_ω; если выполняется $T_j(L_\omega)\, u, v$, то в зависимости от содержимого L_ω принимается решение: выполнять в качестве следующего u-й или v-й оператор. Если на различных шагах машина выполняет $T_j(L_\omega)\, u, v$ и $T_j(L_{\omega'})\, u', v'$ при одном и том же содержимом ячеек L_ω и $L_{\omega'}$, то на первом шаге переход к оператору S_u (u-й оператор) следует разрешать в том и только том случае, если на втором шаге осуществляется переход к оператору $S_{u'}$: иными словами, решения (относительно выбора следующего оператора) должны

быть согласованными. Если процесс вычисления останавливается (при выполнении оператора СТОП), то его значением называется последний вектор значений, присвоенных ячейкам $L_1, \ldots, L_n$. Выполненная последовательность вычислений в общем случае зависит как от интерпретации, так и от природы индивидуальных шагов. Необходимо заметить, что схемы программ предназначены для изображения неинтерпретированных машинных программ в том смысле, что нас интересует их поведение над очень широким классом интерпретаций.

Схема, изображенная на диаграмме I, при некоторых ее интерпретациях над целыми числами будет вычислять функцию-факториал; для этого достаточно взять $I(F_1)(x)=1$, $I(F_2)(x, y)= = x \times y$, $I(F_3)(x) = x - 1$, $I(T_1)(x) = 0$, если $x = 0$, и $I(T_1)(x) = 1$, если $x > 0$. При других интерпретациях, а именно над списочными структурами, она будет вычислять функцию *reverse*(x), рассмотренную Мак-Карти в [4]; для этого надо положить $I(F_1)(x) = \mathrm{NIL}$, $I(F_2)(x, y) = \mathrm{Cons}(\mathrm{Car}(x), y)$, $I(F_3)(x) = \mathrm{Cdr}(x)$ и $I(T_1)(x)$ есть 0, если $\mathrm{Null}(x)$, и I в противном случае.

Формально, *интерпретация* I схемы P состоит в сопоставлении ячейкам, функциональным и предикатным символам схемы P соответственно некоторого множества D и множества функций и характеристических функций, причем

(i) Каждому символу ячейки L_i сопоставляется элемент $I(L_i) \in D$.

(ii) Каждому функциональному символу F_i^n сопоставляется n-арная функция $I(F_i^n)$: $D^n \to D$.

(iii) Каждому предикатному символу T_i сопоставляется характеристическая функция над $DI(T_i)$: $D \to \{0, 1\}$.

Обозначения

Пусть σ и $P(\sigma)$ обозначают соответственно допустимую последовательность операторов схемы P (т. е. путь в потоковой диаграмме схемы P) и последовательность векторов значений, присваиваемых ячейкам $\langle L_1, L_2, \ldots, L_n\rangle$ в процессе выполнения σ. Пусть $\sigma(i)$ — i-й член σ, $P(\sigma)(i)$ — вектор, полученный по выполнении $\sigma(i)$, и $P(\sigma)(i, j)$ — его j-я компонента. Символом S_u обозначим u-й оператор схемы P. Последовательности σ_I и $P_I(\sigma_I)$, соответствующие данной интерпретации I и называемые последовательностями выполнения и вычисления, определим по индукции следующим образом:

(i) $\sigma_I(1) = S_1$.

(ii) $P_I(\sigma_I)(0) = \langle I(L_1), \ldots, I(L_n)\rangle$.

(iii) Если $\sigma_I(i+1)$ есть присваивание

$$k,\ L_\varphi := F_t^v(L_u, \ldots, L_{u_v}),$$

то

$$\sigma_I(i+2) = S_{k+1},$$

$$P_I(\sigma_I)(i+1, j) = P_I(\sigma_I)(i, j), \quad \text{если} \quad j \neq \omega,$$

и

$$P_I(\sigma_I)(i+1, \omega) = I(F_i^v)(P_I(\sigma_I)(i, u_1), \ldots, P_I(\sigma_I)(i, u_v)).$$

(iv) Если $\sigma_I(i+1)$ есть переход

$$T_j(L_\omega)\, u,\ v,$$

то

$$\sigma_I(i+2) = S_u, \quad \text{если} \quad I(T_j)(P_I(\sigma_I)(\mathrm{i}, \omega)) = 0,$$

$$\sigma_I(i+2) = S_v, \quad \text{если} \quad I(T_j)(P_I(\sigma_I)(i, \omega)) = 1$$

и

$$P_I(\sigma_I)(i+1) = P_I(\sigma_I)(\mathrm{i}).$$

Если σ_I конечна (например, длины m), то *значением* вычисления схемы P при I (обозначаем через $\mathrm{Val}(P_I)$) назовем последний вектор последовательности $P_I(\sigma_I)$, т. е. $\mathrm{Val}(P_I) = P_I(\sigma_I)(m)$.

Если последовательность σ выполнима при некоторой интерпретации, то назовем ее *выполнимой последовательностью.*

Рассмотрим интерпретацию I, определенную над классом строк функциональных символов и символов ячеек из P, такую, что $I(L_i) = L_i$ и, если $\alpha_1, \ldots, \alpha_v$ — некоторые строки, то $I(F_i^v)(\alpha_1, \ldots, \alpha_v) = F_i^v \alpha_1 \ldots \alpha_v$, т. е. результату конкатенации строк $F_i^v, \alpha_1, \ldots, \alpha_v$. Такую интерпретацию будем называть *свободной.*

Легко показать, что любая выполнимая последовательность σ выполняется при некоторой свободной интерпретации; для этого рассмотрим ассоциированную с ней свободную последовательность вычислений $P(\sigma)$:

(i) $P(\sigma)(0) = \langle L_1, \ldots, L_n \rangle$.

(ii) Если $\sigma(i+1)$ есть присваивание $L_\omega := F_i^v(L_{u_1}, \ldots, L_{u_v})$, то $P(\sigma)(i+1, j) = P(\sigma)(i, j)$ при $j \neq \omega$ и $P(\sigma)(i+1, \omega)$ берется равным строке, полученной конкатенацией строк $F_i^v, P(\sigma)(i, u_1), \ldots, P(\sigma)(i, u_v)$.

С каждым предикатным символом T_j из P будем связывать два множества: $\mathscr{L}_j(\sigma)$ в $\mathscr{R}_j(\sigma)$; по определению для всякого i, для которого $\sigma(i)$ — это оператор вида $T_j(L_\omega)\, u, v$ с $u \neq v$, $P(\sigma)(i, \omega) \in \mathscr{L}_j(\sigma)$ в том и только том случае, если $\sigma(i+1)$ есть S_v. Поскольку σ — выполнимая последовательность, $\mathscr{L}_j(\sigma) \cap \bigcap R_j(\sigma) = \varnothing$. Следовательно, существует такая свободная

интерпретация I, что при $\alpha \in \mathscr{L}_j(\sigma)$ $I(T_j)(\alpha) = 0$ и при $\alpha \in \mathscr{R}_j(\sigma)$ $I(T_j)(\alpha) = 1$. Очевидно, что $\sigma_I = \sigma$. Условимся обозначать значение последовательности σ при свободной интерпретации через $\mathrm{Val}(P(\sigma))$.

Определение 1. Схемы программ P и P' *строго эквивалентны* ($P \equiv P'$) в том и только том случае, если при всех интерпретациях I равенство $\mathrm{Val}(P_I) = \mathrm{Val}(P'_I)$ имеет место всякий раз, как определено одно из этих значений.

По-видимому, это самое строгое возможное понятие эквивалентности и интуитивно может быть представлено как требование того, чтобы эквивалентные схемы программ вели себя одинаково на всех идеализированных машинах. Можно спорить по поводу чрезмерной ограничительности этого понятия, хотя легко усмотреть, что оно совпадает с понятием, полученным при рассмотрении только свободных интерпретаций или (при проведении гёделевской нумерации формальных строк) интерпретаций над множеством натуральных чисел. Можно получить более «конструктивно приемлемые» понятия эквивалентности, если в определении I ограничить множество интерпретаций. Среди таких понятий мы упомянем *конечную эквивалентность* ($P \equiv {}_F P'$, если $P = P'$ при всех интерпретациях с конечными областями)[1]) и *рекурсивную эквивалентность* ($P \equiv {}_R P'$, если $P \equiv P$ при всех таких интерпретациях, в которых область и функции общерекурсивны). Наконец, при обсуждении проблемы упрощения может представить интерес понятие *слабой эквивалентности: P* и P' слабо эквивалентны ($P \simeq P'$) в том и только том случае, если при всех интерпретациях I равенство $\mathrm{Val}(P_I) = \mathrm{Val}(P'_I)$ выполняется всякий раз, когда определены *оба* значения. (Заметим, что слабая эквивалентность не является отношением эквивалентности.) До определенного момента мы ограничимся рассмотрением строгой эквивалентности.

Существует простая синтаксическая характеристика строгой эквивалентности, выраженная в условиях на последовательности выполнений схем P и P'. Множество $\{\sigma_i\}$ последовательностей, построенных для одной схемы или нескольких схем, называется *согласованным*, если для всякого T_j, входящего в схему (в схемы),

$$\bigcup_i \mathscr{L}_j(\sigma_i) \cap \bigcup_i \mathscr{R}_j(\sigma_i) = \varnothing.$$

Так, σ является выполнимой последовательностью в том и только том случае, если одноэлементное множество $\{\sigma\}$ согласовано.

[1]) Впредь называемых *конечными интерпретациями*.

Теорема 2.1. *$P \equiv P'$ тогда и только тогда, когда для всех согласованных пар последовательностей σ в P и σ' в P' равенство* $\mathrm{Val}(P(\sigma)) = \mathrm{Val}(P'(\sigma'))$ *имеет место всякий раз, как определено одно из этих значений.*

Доказательство. ($\Rightarrow$). Предположим, что $P \equiv P'$ и что σ и σ' согласованы.

Согласованность σ и σ' означает, что существует единственная свободная интерпретация I над областью D, состоящей из компонент векторов, принадлежащих $P(\sigma)$ и $P'(\sigma')$, такая, что $\sigma_I = \sigma$ и $\sigma'_I = \sigma'$. (Просто определим $I(T_j)$ для любого $\alpha \in D$ следующим образом: $I(T_j) = 0$ при $\alpha \in \mathscr{L}_j(\sigma) \cup \mathscr{L}_j(\sigma')$ и $I(T_j) = 1$ в противном случае.) Так как $\mathrm{Val}(P_I) = \mathrm{Val}(P'_I)$, если одно из этих значений определено, то отсюда следует, что либо $\mathrm{Val}(P(\sigma)) = \mathrm{Val}(P'(\sigma'))$, либо оба значения не определены.

($\Leftarrow$). Если I — некоторая интерпретация схем P и P', то пара $\{\sigma_I, \sigma'_I\}$ согласована; следовательно, $\mathrm{Val}(P(\sigma_I)) = \mathrm{Val}(P'(\sigma'_I))$, если одно из значений определено. Но значение $\mathrm{Val}(P_I)$ представляет собой интерпретацию значения $\mathrm{Val}(P(\sigma_I))$ (т. е., если α — компонента вектора из $P(\sigma_I)$, имеющая вид $F_i^m F_j^n \ldots L_k \ldots$, то ее интерпретация есть $I(\alpha) = I(F_i^m) \cdot I(F_j^n)(\ldots I(L_k \ldots)$, и если $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ — вектор из $P(\sigma_I)$, то его интерпретация есть $I(\alpha) = \langle I(\alpha_1), \ldots, I(\alpha_n) \rangle$). Поэтому $\mathrm{Val}(P_I) = \mathrm{Val}(P'_I)$, если одно из значений определено.

Можно было бы надеяться, что строгая эквивалентность будет обладать свойством «компактности» (т. е. что конечная эквивалентность и строгая эквивалентность будут определять одно и то же отношение на схемах); однако это не так.

Теорема 2.2. $P \equiv_F P' \not\Rightarrow P \equiv P'$.

Доказательство. Пусть P — это схема, изображенная диаграммой 2, и P' — простая цепочка

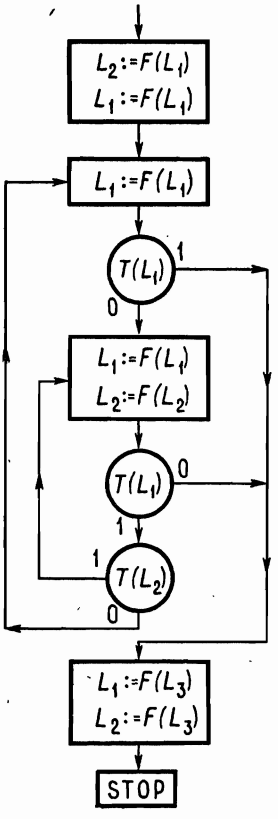
1. $L_1 := F(L_3)$,

2. $L_2 := F(L_3)$,

3. STOP.

Прямая индукция по аргументу устанавливает, что если σ_I — бесконечная последовательность выполнения схемы P, то значениями $I(T)$ на последовательности выражений F^2L_1, F^3L_1, F^4L_1, ... будут соответственно числа 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, ...

(через F^nL_1 обозначается конкатенация F и $F^{n-1}L_1$). Поэтому I не может быть конечной интерпретацией. А на всех конечных интерпретациях P заканчивает свою работу с теми же значениями, что и P'. Итак, $P \equiv {}_FP'$, но $P \not\equiv P'$.



Определенные выше четыре отношения на схемах могут быть выстроены в последовательность строго возрастающей силы.

Теорема 2.3.

$$\simeq \supsetneq \equiv_F \supsetneq \equiv_R \supsetneq \equiv$$

Доказательство. Первое строгое включение очевидно, второе следует из доказательства теоремы 2.2 и третье доказано в [5].

В заключение отметим, что, хотя нами принят алголо-подобный формализм, схемы программ можно задавать посредством систем рекурсивных уравнений без данных (базисных) функций, например посредством систем, рассмотренных Мак-Карти в [4], а поэтому результаты о неразрешимости, полученные в разд. 4, остаются верными и для последнего формализма. Пусть задана схема P с n ячейками $L_1, \ldots, L_n$; построим систему $E(P)$ рекурсивных уравнений от n переменных $\bar{X} = \langle x_1, \ldots, x_n \rangle$. Сначала для каждого функционального символа F из P введем функциональный символ f_k и для каждого предикатного символа T_k — предикатный символ P_k. $E(P)$ будет состоять из уравнений, определяющих функции $g_{i1}, \ldots, g_{in}$, которые строятся для каждого оператора S_i следующим образом: если S_i — присваивание

$$i.L_\omega := F_k(L_{u_1}, \ldots, L_{u_v}),$$

то $E(P)$ содержит уравнения

$$g_{ij}(\bar{X}) = g_{i+1,j}(x_1, \ldots, x_{\omega-1}, f_k(x_{u_1}, \ldots, x_{u_v}), \ldots, x_n), \quad j \leqslant n;$$

для всякого перехода

$$i.T_k(L_\omega)\, u, v$$

$E(P)$ содержит условные формулы

$$g_{ij}(\bar{X}) = (P_k(x_\omega) \to g_{uj}(\bar{X}),\ T \to g_{vj}(\bar{X})), \quad j \leqslant n.$$

Если S_i — оператор STOP, то соответствующие уравнения имеют вид $g_{ij}(\bar{X}) = x_j$, $j \leqslant n$. Значения функций $g_{i1}(\bar{X}), \ldots, g_{in}(\bar{X})$, если они определены, будут равны последним значениям, присвоенным ячейкам $L_1, \ldots, L_n$ соответственно в останавливающемся вычислении схемы P, начатом с оператора S_i при такой интерпретации I, что $I(L_j) = x_j$, $I(F_k) = f_k$ и $I(T_k) = P_k$. Легко видеть, что $P \equiv P'$ в том и только том случае, если при всех $i \leqslant n$ выполняется равенство $g_{1i} = g'_{1i}$.

3. МНОГОЛЕНТОЧНЫЕ И МНОГОГОЛОВОЧНЫЕ АВТОМАТЫ

Розенберг в [7] и независимо от него авторы установили ряд результатов по неразрешимости, включая результат для автоматов с двумя головками. Оказалось, что для моделирования этих автоматов в известном смысле могут быть использованы схемы с двумя ячейками, и мы воспользуемся этим обстоятельством, чтобы установить неразрешимость разного вида проблем остановки и проблем эквивалентности для схем программ.

Многоголовочные автоматы, рассмотренные Розенбергом, находятся в тесном отношении с многоленточными односторонними конечными автоматами, введенными Рабином и Скоттом (см. [6], определение 15). Многоголовочный автомат может быть получен из многоленточного автомата требованием, чтобы его ленты были идентичными. Ясно, что такая машина может рассматриваться как одноленточный автомат, головки которого независимо друг от друга считывают ленту и вначале все находятся над исходной позицией ленты.

Многоленточный или многоголовочный автомат A определяется заданием некоторого алфавита $\Sigma = \{s_1, s_2, \ldots, s_n\}$, конечного множества состояний Q и таблицы переходов. Q разбивается на множество *живых* состояний $Q' = \{q_0, q_1, \ldots, q_m\}$ с выделенным начальным состоянием q_0 и множество *мертвых* состояний $Q - Q'$ с выделенным заключительным состоянием a. Q' разбивается на попарно непересекающиеся подмножества Q_j, по одному на каждую считывающую головку (головки находятся на отдельных лентах в случае многоленточного автомата и на одной и той же ленте в случае многоголовочного автомата); если $q_i \in Q_j$, то будем пользоваться записью q_i^j, чтобы зафиксировать, что A, находясь в состоянии q_i, считывает символ j-й головкой. Правила таблицы переходов автомата A имеют вид

$$q_i^j,\ s_1 \to r_{i1},\quad q_i^j,\ s_2 \to r_{i2},\ \ldots,\ q_i^j,\ s_n \to r_{in}$$

для $0 \leqslant i \leqslant m$. Каждое правило определяет следующее состояние $r_{ik} \in Q$, зависящее от символа s_k, считанного головкой j. Прочитав символ, головка j сдвигается к следующему символу на ленте (головка может двигаться только в одном направлении). Достигнув мертвого состояния, автомат «останавливается».

Наши автоматы в одном важном отношении отличаются от автоматов, рассмотренных Рабином и Скоттом или Розенбергом; они оперируют с бесконечными лентами (элементами из Σ^ω, т. е. бесконечными последовательностями над Σ). Для каждой ленты будем различать три возможные ситуации. Если достигнуто a, то лента *принята*. Если достигнуто любое другое мертвое состояние, то лента *отвергнута*. В остальных случаях автомат *расходится* на ленте. Следовательно, автомат A определяет разбиение Σ^ω на три множества: $\mathscr{T}_a(A)$, $\mathscr{T}_r(A)$, $\mathscr{T}_d(A)$ — классы лент, которые A принимает, отвергает и на которых он расходится.

Пусть даны машина Тьюринга M и начальная запись на ленте IT. Существует эффективный метод для построения такого *двухголовочного* автомата A, который проверяет, описывает ли его лента вычисление, выполняемое машиной M на IT. Лента принимается, если ее начальный сегмент, условно выражаясь, «описывает» завершенное машиной M вычисление на IT; A расходится, если лента «описывает» незавершающееся вычисление, реализуемое машиной M на IT; в остальных случаях автомат A достигает некоторого мертвого состояния r, и лента отвергается. Чтобы в подразумеваемом здесь смысле «описывать» вычисление, сегмент ленты должен регистрировать последовательность закодированных конфигураций (по Дэвису, см. [1], «мгновенных описаний»), составленных для последовательных шагов вычисления и удовлетворяющих (ради дальнейшего удобства) следующим условиям:

(а) Начальная конфигурация, составленная для IT, кодируется таким образом, что символ начального состояния не является ни первым, ни последним символом.

(б) Каждая последующая конфигурация получается путем приписывания «пустого» символа к обоим концам записи, которая является результатом применения одного из правил машины M к предшествующей конфигурации (и которая совпадает по длине с предшествующей конфигурацией).

(с) Конфигурации отделяются друг от друга некоторым символом β, не принадлежащим алфавиту машины M.

Работа автомата A определяется следующим образом (для альтернативного описания см. теорему 9(а) в работе Розенберга [7]): сначала A с помощью головки 1 проверяет начальную конфигурацию; затем A последовательно сравнивает каждую кон-

фигурацию с ей предшествующей, используя головку 1 для считывания следующей конфигурации, в то время как головка 2 считывает предыдущую. После каждого сравнения головки устанавливаются точно в те позиции, из которых начинается следующее сравнение. Сравнение двух конфигураций протекает следующим образом: головкой 1 считывается пустой символ; затем по очереди сравниваются символы из двух конфигураций и так до тех пор, пока одной из головок не будет прочитан символ состояния; тогда в каждой из двух конфигураций считывается следующая пара символов. Если полученные тройки символов не соответствуют подходящему правилу машины M, то лента отвергается; например, если Q_j, Q_l — символы состояний машины M, то пара троек должна иметь вид $\langle Q_j s_k s_p,\ s_k Q_l s_p\rangle$, или вид $\langle s_p Q_j s_k,\ Q_l s_p s_k\rangle$, или вид $\langle Q_j s_k s_p,\ Q_l s_i s_p\rangle$ в зависимости от того, будет ли соответствующая четверка (в обозначениях Дэвиса, см. [1]) иметь вид $Q_j s_k R Q_l$, или вид $Q_j s_k L Q_l$, или вид $Q_j s_k s_i Q_l$. Если тройка, прочитанная головкой 1, свидетельствует о том, что рассматриваемая конфигурация заключительная, то лента принимается; в противном случае вновь сравниваются символы двух конфигураций и так до тех пор, пока головкой 2 не будет прочитан символ β; тогда головка 1 считывает пустой символ, затем символ β и возобновляется процесс сравнения. В течение описанного процесса может встретиться лишь фиксированное конечное число ситуаций, подлежащих различению. Итак, по данным M и IT мы можем построить конечный автомат A, осуществляющий описанное сравнение. Более того, это построение является эффективным. Дальнейшие детали нами опускаются.

Автомат A обладает свойствами:

$$\mathscr{T}_a(A) = \varnothing \Leftrightarrow \mathscr{T}_d(A) \neq \varnothing$$

$\Leftrightarrow M$ не останавливается при работе с IT.

Хорошо известно, что последнее свойство не является частично разрешимым. Это приводит к следующей теореме.

Теорема 3.1. *Для двухголовочных автоматов A свойства*

$$\text{(а)}\ \mathscr{T}_a(A) = \varnothing,$$

$$\text{(б)}\ \mathscr{T}_d(A) \neq \varnothing$$

не являются частично разрешимыми.

Моделирование автоматов схемами особенно легко осуществить в случае бинарных автоматов, т. е. автоматов над алфавитом {0, 1}. Но сначала для бинарных автоматов установим результат, подобный теореме 3.1.

Обозначение. Для любого $i \geqslant 0$ символом $\underline{i}$ будем обозначать строку, состоящую из i вхождений символа «1», замыкаемых одним символом «0».

Обозначим через $\mathscr{B}$ преобразование, переводящее строки, ленты, множества лент и другие объекты над алфавитом $\Sigma = \{s_1, s_2, \ldots, s_n\}$ в бинарные строки, ленты и другие соответствующие объекты и индуцируемое естественным образом применением следующих правил:

$$\mathscr{B}(s_i) = \underline{i},$$

$$\mathscr{B}(s_i w) = \underline{i}\,\mathscr{B}(w) \text{ для строк вида } s_i w$$

и т. д.

Лемма 3.2. *Существует такое эффективное преобразование* $\overline{\mathscr{B}}$ *автоматов над алфавитом* Σ *в бинарные автоматы, что*

(a) $\mathscr{T}_a(\overline{\mathscr{B}}(A)) = \varnothing \Leftrightarrow \mathscr{T}_a(A) = \varnothing$,

(b) $\mathscr{T}_d(\overline{\mathscr{B}}(A)) = \mathscr{B}(\mathscr{T}_d(A))$.

Доказательство. Пусть правила таблицы, определяющей автомат A, имеют вид

$$q_i^l,\ s_1 \to r_{i1},\ q_i^l,\ s_2 \to r_{i2},\ \ldots,\ q_i^l,\ s_n \to r_{in}.$$

Таблица для автомата $\overline{\mathscr{B}}(A)$ получается заменой каждого правила таблицы $An + 1$ правилами

$$q_i^l,\ 0 \to \underline{r'},\quad q_i^l,\ 1 \to q_{i1}^l,$$

$$q_{i1}^l,\ 0 \to r_{i1},\quad q_{i1}^l,\ 1 \to q_{i2}^l,$$

$$q_{i2}^l,\ 0 \to r_{i2},\quad q_{i2}^l,\ 1 \to q_{i3}^l,$$

$$\ldots$$

$$q_{in}^l,\ 0 \to r_{in},\quad q_{in}^l,\ 1 \to \underline{r''},$$

где q_{ik}^l, $1 \leqslant k \leqslant n$, — новые состояния, не входящие в таблицу автомата A, а r', r'' — добавочные мертвые состояния. Остальные живые и мертвые состояния автомата $\overline{\mathscr{B}}(A)$ те же самые, что и у автомата A.

Пусть автомат A, головки которого установлены над позициями t_1 и t_2 его ленты, из состояния q_i непосредственно переходит в состояние r_{ik}, причем его головки устанавливаются над позициями t_1' и t_2'. Очевидно, что это будет иметь место в том и только том случае, если автомат $\overline{\mathscr{B}}(A)$, головки которого установлены над позициями $\mathscr{B}(t_1)$ и $\mathscr{B}(t_2)$, из состояния q_i, пройдя последовательно через $k + 1$ состояний, переходит в состояние r_{ik}, причем его головки устанавливаются над позициями $\mathscr{B}(t_1')$

и $\mathscr{B}(t_2')$. Итак, индукцией по числу пройденных автоматом A состояний доказывается, что A принимает ленту t или расходится на ней в том и только том случае, если $\overline{\mathscr{B}}(A)$ принимает ленту $\mathscr{B}(t)$ или расходится на ней. Следовательно, $\mathscr{T}_d(\overline{\mathscr{B}}(A)) \supseteq \supseteq \mathscr{B}(\mathscr{T}_d(A))$ и $\mathscr{T}(\overline{\mathscr{B}}_a(A)) \supseteq \mathscr{B}(\mathscr{T}_a(A))$.

С другой стороны, если бинарная лента t' не отвергается автоматом $\overline{\mathscr{B}}(A)$, то ни одно из состояний r' и r'' не достижимо, и поэтому начальный сегмент, считанный автоматом, является начальным сегментом некоторой ленты $\mathscr{B}(t)$. Если $\overline{\mathscr{B}}(A)$ принимает t', то $\overline{\mathscr{B}}(A)$ принимает и ленту $\mathscr{B}(t)$, которая отличается от t' только несчитанными позициями; следовательно, A принимает t. Поэтому $\mathscr{T}_a(\overline{\mathscr{B}}(A)) \neq \varnothing \Rightarrow \mathscr{T}_a(A) \neq \varnothing$. Если $\overline{\mathscr{B}}(A)$ расходится на t', то $t' = \mathscr{B}(t)$ и A расходится на t. Следовательно, $\mathscr{T}_d(\overline{\mathscr{B}}(A)) \subseteq \mathscr{B}(\mathscr{T}_d(A))$. Лемма 3.2 доказана.

4. ОСНОВНЫЕ РЕЗУЛЬТАТЫ ПО НЕРАЗРЕШИМОСТИ

Теперь можно описать метод моделирования схемами двухголовочных бинарных автоматов. Для этих целей достаточно рассмотреть схемы, содержащие один унарный функциональный символ F, один предикатный символ T и ячейки $L_1, L_2, \ldots$. Понятие «интерпретация», «выражение» и т. д. надо понимать теперь как ограничения соответствующих понятий, вызванные рассмотрением таких схем.

Обозначение. Символом $\mathscr{S}_i$ обозначим класс схем, содержащих не более i ячеек $L_1, L_2, \ldots L_i$.

Определение. Пусть E — некоторое выражение и I — некоторая интерпретация. Назовем I — лентой для E бинарную ленту

$$t_I(E) = \varepsilon_0\varepsilon_1\varepsilon_2 \ldots,$$

где

$$\varepsilon_0 = T_I(I(E))$$

и

$$\varepsilon_i = T_I(F_I^i(I(E))), \qquad i \geqslant 1.$$

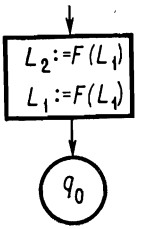
Описываемая ниже процедура строит по заданному бинарному двухголовочному автомату B неполную схему $P(B)$, в которой каждое живое состояние автомата B представлено парой ему соответствующих операторов (присвоением и следующим за ним переходом). Схема $P(B)$ неполна потому, что некоторые из ее адресов перехода изображаются символическими метками, которые подлежат определению при ее достраивании. Какой бы ни была интерпретация I, $P(B)$ моделирует поведение автомата B на ленте $t_I(F(L_1))$ в следующем смысле. Вычисление,

осуществленное автоматом B, описывается последовательностью $q_0\varepsilon_1 q_{i_1}\varepsilon_2 q_{i_2}\varepsilon_3 \ldots$, состоящей из пар «состояние — считываемый символ», в том и только том случае, если последовательность выполнения σ_I схемы $P(B)$ совпадает с последовательностью пар операторов, которые соответствуют состояниям q_0, q_{i_1}, q_{i_2}, ... Таким образом, в схеме $P(B)$ достигается метка, соответствующая мертвому состоянию, если последнее достижимо автоматом B, и схема $P(B)$ расходится [1]) в том и только том случае, когда B расходится на этой ленте. Включением $P(B)$ в другие схемы будут получены преобразования автоматов в схемы, сохраняющие различные отношения эквивалентности.

Предположим, что правила таблицы, определяющей автомат B, имеют вид

$$q_i^l,\ 0 \to r_{i0},\quad q_i^l,\ 1 \to r_{i1},$$

где $0 \leqslant i \leqslant N$. Тогда схема $P(B)$ получается из $(N+2)$-х фрагментов, использующих состояния автомата B в качестве меток и имеющих следующий вид:



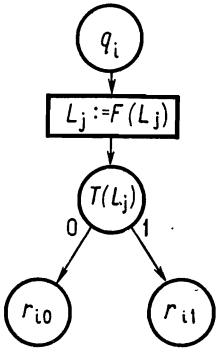
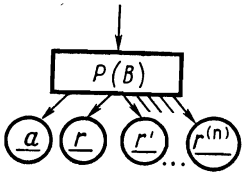


Схема $P(B)$ представляет собой результат совмещения меток в вышеописанных фрагментах и содержит «метки выходов», соответствующие мертвым состояниям автомата B.

Если мертвые состояния автомата B — это a, r, r', ..., $r^{(n)}$, то результат этого процесса объединения фрагментов будем изображать в виде



[1]) Под расходимостью схемы понимается бесконечность процесса ее выполнения. — *Прим. перев.*

Индукцией по j легко показать, что если по выполнении $j+2$ актов присваивания в схеме $P(B)$ достигается метка q_i и при этом ячейки L_1 и L_2 «хранят» выражения E_1 и E_2 соответственно, то по выполнении j переходов из состояния в состояние, осуществленных автоматом B на ленте $t_I(F(L_1))$, он находится в состоянии q_i и его головки находятся над позициями $t_I(E_1)$ и $t_I(E_2)$ соответственно. Следовательно, $P(B)$ моделирует работу автомата B на ленте $t_I(F(L_1))$ в смысле, в общих чертах описанном выше.

С другой стороны, какой бы ни была бинарная лента $t = \varepsilon_1\varepsilon_2 \ldots$, заданная для B, схема $P(B)$ будет моделировать работу автомата B на этой ленте, если выбрать интерпретацию I, обладающую свойством $t = t_I(F(L_1))$. Чтобы последнее выполнялось, достаточно взять в качестве I свободную интерпретацию (областью определения которой является класс выражений), для которой $T_I(F^{i+1}L_1) = \varepsilon_i$, $i \geqslant 1$. Следовательно, все вычисления автомата B и только они моделируются схемой $P(B)$.

Обозначение. Обозначим через D простую схему

1. $T(L_1)$ 1, 1

и через Z — схему

1. L_1: $= F(L_3)$,
2. L_2: $= F(L_3)$,
3. STOP.

Теорема 4.1. *Для схем P из $\mathscr{S}_2$ не являются частично разрешимыми следующие свойства:*

(a) *P расходится на всех интерпретациях.*
(b) *P расходится на всех конечных интерпретациях.*
(c) *P расходится хотя бы на одной интерпретации.*
(d) *P останавливается* [1]) *на всех конечных интерпретациях.*
(e) $P \equiv D$.
(f) $P \equiv {}_F D$.

Для схем P из $\mathscr{S}_3$ не имеет места частичная разрешимость следующих свойств:

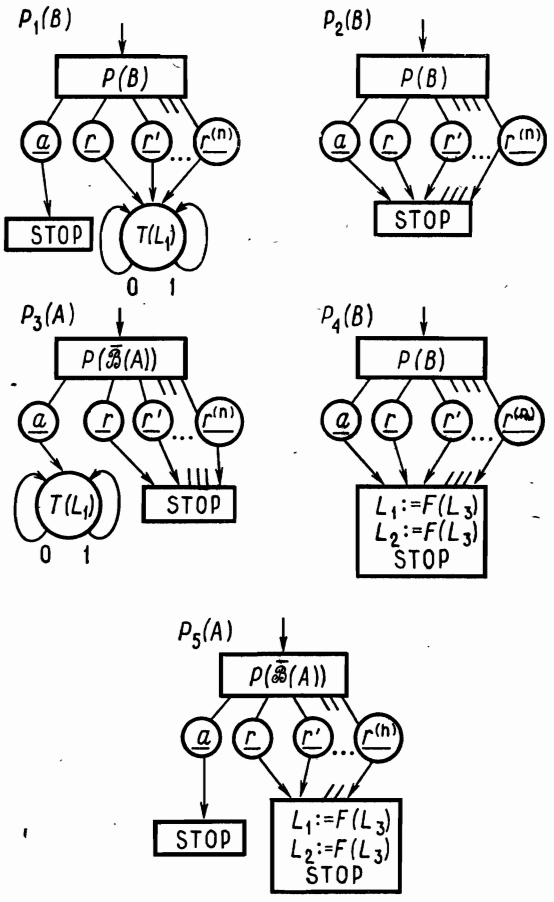
(g) $P \not\equiv Z$,
(h) $P \equiv {}_F Z$,
(i) $P \simeq Z$.

Доказательство. (a) Рассмотрим схему $P_1(B)$, представленную на диаграмме 3. Она эффективно строится по B и расходится при всех интерпретациях тогда и только тогда, когда $\mathscr{T}_a(B) = \varnothing$. Утверждаемый результат следует из 3.1(a) и 3.2(a).

[1]) Под остановкой схемы понимается завершение процесса ее выполнения. — *Прим. перев.*

(b) эквивалентно утверждению (a). Если P когда-либо останавливается, то это произойдет и на некоторой конечной интерпретации.

(c) Схема $P_2(B)$, изображенная на диаграмме 3, эффективно строится по B и расходится хотя бы на одной интерпретации в том и только том случае, если $\mathcal{I}_d(B) \neq \varnothing$. Утверждаемый результат следует из 3.1(b) и 3.2(b).



(d) Вспомним автомат A, построенный по машине Тьюринга и ее начальной конфигурации, причем эффективно, для доказательства теоремы 3.1. A может расходиться только при условии, что его лентой описывается незавершающееся вычисление, выполняемое машиной Тьюринга. Заметим, что A не будет расхо-

диться на ленте, которая, начиная с некоторого места, является периодической, так как при расходимости число символов между последовательными вхождениями разделительного символа β неограниченно растет. Следовательно, A останавливается на каждой, начиная с некоторого места, периодической ленте.

Теорема 3.1, как легко видеть, остается верной, если ограничиться рассмотрением только автоматов, обладающих описанным выше свойством. Преобразование $\overline{\mathscr{B}}$, применяемое в 3.2, сохраняет это свойство автомата A. Наконец, заметим, что если I — конечная интерпретация, то при любом E лента $t_I(E)$, начиная с некоторого места, периодична, так как при некоторых $i \neq j$ $F_I^i(I(E)) = F_I^j(I(E))$.

Итак, если A обладает описанным выше свойством, то $P(\overline{\mathscr{B}}(A))$ может разойтись лишь на интерпретации, не являющейся конечной. Кроме того, если метка a может быть достигнута в этой схеме, то она может быть достигнута на некоторой конечной интерпретации. Следовательно, если A останавливается на любой, начиная с некоторого места, периодической ленте, то схема $P_3(A)$, представленная на диаграмме 3, останавливается на всех конечных интерпретациях в том и только том случае, если $\mathscr{I}_a(\overline{\mathscr{B}}(A)) = \varnothing$. Таким образом, утверждаемый результат следует из теоремы 3.1, рассматриваемой для описанного выше класса автоматов, и из 3.2(a).

(e), (f) эквивалентны утверждениям (a), (b).

(g) Схема $P_4(B)$ эффективно строится по B и $P_4(B) \not\equiv Z \Leftrightarrow \Leftrightarrow \mathscr{I}_d(B) \neq \varnothing$. Поэтому рассматриваемое утверждение устанавливается так же, как и утверждение (c).

(h), (i). Соображение, использованное при доказательстве (d), применим к схеме $P_5(A)$; $P_5(A) \simeq Z \Leftrightarrow P_5(A) \equiv_F Z \Leftrightarrow \Leftrightarrow \mathscr{I}_a(\overline{\mathscr{B}}(A)) = \varnothing$. Если метка a может быть достигнута, то она может быть достигнута на конечной интерпретации, которая нарушает эквивалентность ($\equiv_F$ или $\simeq$) с Z при подходящим образом выбранном значении $I(L_3)$.

Заметим, что $P \simeq D$ всегда верно, следовательно, разрешимо. Заслуживает внимания тот факт, что свойства, дополнительные к тем, что рассматривались в 4.1, все являются частично разрешимыми. Имеются в виду следующие результаты, справедливые для схем P без каких-либо ограничений.

Теорема 4.2. *Пусть Q — схема, которая останавливается при всех интерпретациях, и R — произвольная схема. Имеет место частичная разрешимость следующих свойств схемы P.*

(a) *P останавливается хотя бы на одной интерпретации.*

(b) *P останавливается хотя бы на одной конечной интерпретации.*

(c) *P останавливается на всех интерпретациях.*
(d) *P расходится хотя бы на одной конечной интерпретации.*
(e) $P \not\equiv D$.
(f) $P \not\equiv {}_{F}D$ (*подслучай случая* (h)).
(g) $P \equiv Q$.
(h) $P \not\equiv {}_{F}R$.
(i) $P \not\simeq R$.

Доказательство (в общих чертах). На конечной интерпретации I, насчитывающей n элементов, схема P может рассматриваться как машина с конечным числом, а именно $p \cdot n^k$, состояний, где k — число ячеек и p — число операторов в схеме P. Каждое свойство, за исключением (c) и (g), может быть проверено последовательным просмотром класса конечных интерпретаций до тех пор, пока не будет найдена интерпретация с подходящим свойством. Проверки, выполняемые для каждой конечной интерпретации, являются рекурсивными, в силу хорошо известных свойств машин с конечным числом состояний Проверки, необходимые при установлении (a), (b), (d), (e), (f),— это проверки на остановку или расходимость, при установлении (h), (i) — это проверки на «строгую» и «слабую» эквивалентность соответственно машин с конечным числом состояний (эти понятия определяются по аналогии с соответствующими понятиями, введенными выше).

Чтобы проверить свойства (c) и (g), необходимо эффективно перечислить все возможные последовательности выполнения схемы P. Если P сходится на всех интерпретациях, этот процесс будет конечным (можно просматривать конечные пути, начинающиеся во входе схемы P, отбрасывая те из них, которые, каждый в отдельности, являются несогласованными в смысле, определенном в разд. 2, и выбирая для последующего рассмотрения только пути, которые продолжают предварительно оставленные пути; если этот процесс продолжается бесконечно, то в силу того, что выполнены предположения леммы Кёнига, можно утверждать, что существует бесконечный согласованный путь, и, следовательно, по результатам разд. 2, имеет место расходимость. Свойство (c), таким образом, установлено. Для того чтобы установить свойство (g), необходимо проверять, совпадают ли окончательные выражения, полученные на согласованных парах последовательностей выполнения схем P и Q.

Заметим, что проблема эквивалентности для класса схем, которые всегда останавливаются, является разрешимой, хотя сама принадлежность этому классу неразрешима.

5. «РАЗУМНЫЕ» ОТНОШЕНИЯ ЭКВИВАЛЕНТНОСТИ

Теорема 4.1, а именно утверждения (e), (f), (h), (i), устанавливают основные результаты относительно частичной неразрешимости эквивалентностей между схемами. В этом разделе получены некоторые более общие результаты. Во-первых (см. 5.1), основной результат обобщен так, чтобы охватить более широкий класс понятий эквивалентности. Во-вторых (см. 5.2), этот результат установлен для более ограниченного класса схем, а именно схем, которые останавливаются на всех конечных интерпретациях.

Представляется разумным принять такое общее определение эквивалентности между схемами, чтобы, с одной стороны, эта эквивалентность имела бы место для схем, которые ведут себя одинаково на всех интерпретациях, конечных и бесконечных, т. е. для строго эквивалентных схем, и, с другой стороны, она бы не выполнялась для схем, которые вырабатывают явно различные результаты на некоторой интерпретации (конечной, если нужно), т. е. для слабо неэквивалентных схем. Примером отношений эквивалентности, не удовлетворяющих этим требованиям, могут служить отношения «доказуемой» эквивалентности (относительно некоторой рекурсивно аксиоматизируемой формальной системы). В частности, «доказуемая» строгая эквивалентность сильнее, чем обычная строгая эквивалентность, так как в противовес утверждению 4.1(e) механизм теорем перечисления, заложенный в формальной системе, позволяет построить процедуру частичного разрешения для отношения $P \equiv D$. Мы здесь не будем привлекать эти понятия.

Определение. Пусть $P \sim Q$ — некоторое отношение между схемами P и Q. Отношение $\sim$ назовем разумным в классе схем $\mathscr{S}$, если для всех P и Q из $\mathscr{S}$

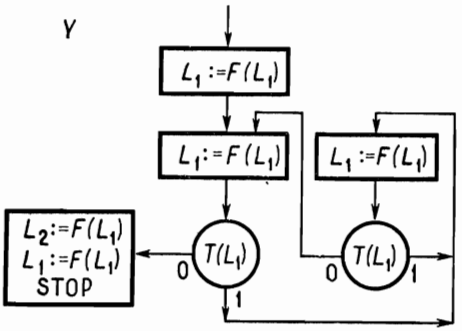
(i) $P \equiv Q \Rightarrow P \sim Q$,

(ii) $P \sim Q \Rightarrow P \simeq Q$.

Заметим, что отношения $\equiv$, $\equiv_R$, $\equiv_F$, $\simeq$ являются разумными в классе всех схем. Разумное отношение не обязано быть симметричным или транзитивным, но должно быть рефлексивным в силу (i). Например, отношение $\simeq$ симметрично, но не транзитивно. Отношение $>$ является транзитивным, но не симметричным; здесь $P > Q$, если всякий раз, как определено $\mathrm{Val}(Q_I)$, определено и $\mathrm{Val}(P_I)$ и имеет место $\mathrm{Val}(P_I) = \mathrm{Val}(Q_I)$.

Определение. Будем говорить, что бинарная лента (или конечная строка) содержит границу, если она начинается символом «0» или содержит два соседних вхождения символа «0».

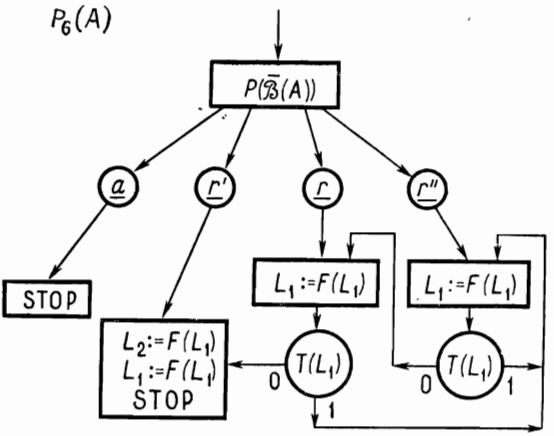
Обозначим через Y следующую схему:



Очевидно, что Y расходится только на таких интерпретациях I, для которых $t_I(F(L_1))$ не содержит границы.

Теорема 5.1. *Если отношение $\sim$ разумно в $\mathscr{P}_2$, то $P \sim Y$ не является частично разрешимым.*

Доказательство. Пусть A — некоторый двухголовочный автомат, построенный в 3.1. Напомним, что A имеет только одно отвергающее состояние $\underline{r}$ и при любом вычислении, проводимом автоматом A, головка 2 никогда не обгоняет головку 1.



Рассмотрим теперь схему $P_6(A)$, эффективно полученную по A ($\underline{r'}$, $\underline{r''}$ — мертвые состояния, введенные при построении $\overline{\mathscr{B}}$ в 3.2). Заметим, что схемы $P_6(A)$ и Y устроены так, чтобы при любой такой интерпретации I, для которой автомат $\overline{\mathscr{B}}(A)$ не принимает ленты $t_I(F(L_1))$, они вырабатывают одинаковые результаты или одновременно расходятся, если лента $t_I(F(L_1))$ не содержит границы. Отметим, что состояние $\underline{r'}$ достигается, если

вычисление, выполняемое автоматом $\overline{\mathfrak{B}}(A)$, обрывается неожиданно [1]), если достигнуто состояние $\underline{r}$, то предшествующим символом был символ «0», или вообще не был прочитан никакой символ; если достигнуто состояние $\underline{r''}$, то предшествующим символом был символ «1». Следовательно,

$$\mathfrak{I}_a(\overline{\mathfrak{B}}(A)) = \varnothing \Rightarrow P_6(A) \equiv Y. \tag{1}$$

С другой стороны, предположим, что $\overline{\mathfrak{B}}(A)$ принимает некоторую ленту t. Возьмем такую свободную интерпретацию I, что лента $t_I(F(L_1))$ совпадает с t на сегменте, считанном автоматом $\overline{\mathfrak{B}}(A)$, и содержит границу (в некотором месте за пределами сегмента). Тогда оба значения $\mathrm{Val}(Y_I)$ и $\mathrm{Val}(P_6(A)_I)$ определены и $\mathrm{Val}(Y_I) \neq \mathrm{Val}(P_6(A)_I)$. Следовательно,

$$\mathfrak{I}_a(\overline{\mathfrak{B}}(A)) \neq \varnothing \Rightarrow P_6(A) \not\equiv Y. \tag{2}$$

Но если отношение $\sim$ разумно, то из его определения и из (1) и (2) следуют

$$\mathfrak{I}_a(\overline{\mathfrak{B}}(A)) = \varnothing \Rightarrow P_6(A) \sim Y,$$

$$\mathfrak{I}_a(\overline{\mathfrak{B}}(A)) \neq \varnothing \Rightarrow P_6(A) \not\sim Y.$$

Таким образом, $\mathfrak{I}_a(\overline{\mathfrak{B}}(A)) = \varnothing \Leftrightarrow P_6(A) \sim Y$. Согласно результатам 3.1, 3.2, рассматриваемое свойство автомата A не является частично разрешимым, схема же $P_6(A)$ эффективно строится по A. Отсюда следует утверждение теоремы.

В 5.1 существенно то, что Y — это схема, которая расходится хотя бы на одной интерпретации, так как при условии, что Q всегда останавливается, из 4.2(g) следует частичная разрешимость отношения $P = Q$. Тем не менее заметим мимоходом, что даже в этом случае, каким бы ни было разумное отношение $\sim$, по крайней мере одно из свойств $P \sim Q$, $P \not\sim Q$ не будет частично разрешимым. Доказательство (детали его опущены) осуществляется описанием некоторого эффективного построения по заданному целому x такой схемы P_x, что

$$\varphi_x(x) = 0 \Rightarrow P_x \equiv Q$$

и

$$\varphi_x(x) = 1 \Rightarrow P_x \not\equiv Q;$$

здесь $\varphi_0, \varphi_1, \varphi_2, \ldots$ — гёделевская нумерация всех частично рекурсивных функций. Если $P_x \sim Q$ разрешимо, то существует такая общерекурсивная функция φ_z, что

$$P_x \sim Q \Rightarrow \varphi_z(x) = 1$$

[1]) И обязательно при первой встрече с двумя соседними вхождениями символа «0». — *Прим. перев.*

и

$$P_x \not\equiv Q \Rightarrow \varphi_z(x) = 0.$$

Но тогда $\varphi_z(z) \neq 0 \Longleftrightarrow \varphi_z(z) = 1$. Существенно, что это является доказательством «рекурсивной неотделимости» множеств $\{P/P \equiv Q\}$ и $\{P/P \not\equiv Q\}$, фактически проходящим при любой схеме Q, которая останавливается хотя бы на одной интерпретации (и не проходящим, если Q всегда расходится).

Следующим результатом мы продемонстрируем, что тот факт, что Y расходится на некоторых интерпретациях, не является местом, которое нельзя обойти при усилении утверждения 5.1. Будет показано, что результат, аналогичный 5.1, имеет место для частного класса схем, останавливающихся на всех конечных интерпретациях. Принадлежность этому частному классу является разрешимой. Отсюда следует, что в каком-нибудь классе схем может не существовать такого подкласса, который избежал бы угрозы частичной неразрешимости; по крайней мере, этот выбранный класс не должен содержать схем, которые, среди других их свойств, всегда останавливаются, когда интерпретируемы как программы вычислительных машин.

Теорема 5.2. *Существуют схема $\bar{Y}$ и класс схем $\mathscr{P} \subseteq \mathscr{P}_2$, такие, что*

(i) *$\mathscr{P}$ содержит схемы, которые останавливаются на всех конечных интерпретациях.*

(ii) *$P \in \mathscr{P}$ является разрешимым.*

(iii) *$\bar{Y} \sim P$, где $P \in \mathscr{P}$, не является частично разрешимым ни при каком разумном $\sim$.*

Доказательство. Как и в случае теоремы 5.1, нужно найти автомат $\bar{A}$ и класс автоматов $\{A'\}$, которые могут проверять вычисления, выполняемые машиной Тьюринга, и обладают тем свойством, что любой автомат A' повторяет работу автомата $\bar{A}$, если последний выполняет расходящееся вычисление машины Тьюринга. Дополнительно к этому автомат $\bar{A}$ должен сходиться на всех лентах $t_I(E)$, построенных для конечных интерпретаций I.

При доказательстве 4.1(d) нами отмечено, что автоматы A, используемые в 3.1, обладают этим последним свойством, так как в каждой ленте, на которой A расходится, последовательные сегменты между вхождениями символа β увеличиваются по длине (в точности на два символа); было отмечено также, что преобразование $\overline{\mathscr{B}}$ сохраняет это свойство, и поэтому описанная выше лента не может соответствовать какой-либо конечной интерпретации. Итак, в качестве $\bar{A}$ можно взять автомат, который просматривает вдоль ленты промежутки между вхождениями β, расходится на ней, если промежутки увеличиваются правильно,

и отвергает ее, если это не так. Тем не менее приведем два следующих соображения.

(a) Поскольку автомат $\bar{A}$ имеет конечный алфавит, класс вычислений машины Тьюринга, проверяемый автоматами из $\{A'\}$, должен иметь конечное число состояний и символов и все еще неразрешимую проблему остановки. Это может быть достигнуто сужением этого класса до вычислений, выполняемых некоторой фиксированной универсальной машиной Тьюринга M_0. Тогда алфавит Σ_0 автомата $\bar{A}$ и автоматов из $\{A'\}$ будет состоять из символов, записываемых на ленту, и символов состояний машины M_0, дополненных разделительным символом β. Предположим, что $\Sigma_0 = \{\beta, s_2, s_3, \ldots, s_n\}$. Автомат A' мыслится как такой, который эффективно строится по начальной записи на ленте IT и обладает свойством

$$\mathscr{T}_a(A') = \varnothing \Leftrightarrow M_0 \text{ расходится на } IT.$$

Последнее свойство, как хорошо известно, не будет частично разрешимым.

(b) Начальные конфигурации, проверяемые автоматами A', неограничены по длине, если A' изменяется. При конструировании $\bar{A}$ необходимо избежать того, чтобы он разошелся, скажем, на ленте, не содержащей вхождений β; а такая лента вынудит его разойтись, если только ленты для A' сохранят вид, принятый в конструкции в 3.1. Чтобы обойти эту трудность, подчиним автоматы $\bar{A}$ и A' следующим требованиям.

Для любой строки w над алфавитом Σ_0 символом $l(w)$ будем обозначать ее длину.

(i) $\bar{A}$ должен расходиться только на лентах вида

$$t = w_0\beta w_1\beta w_2\beta \ldots,$$

где w_i — это строки над $\Sigma_0' = \{\beta\}$, а $l(w_i) = 2i + 1$, $i \geqslant 0$. $\bar{A}$ отвергает любую другую ленту.

(ii). По всякой начальной записи IT некоторым эффективным способом будем выбирать для IT начальную конфигурацию w, такую, что $l(w) = 2K + 1$ для подходящего K. Тогда A' должен принимать всякую ленту с начальным сегментом вида

$$w_0\beta w_1\beta \ldots w_{N-1}\beta w_N,$$

где w_i — строки над $\Sigma_0 - \{\beta\}$ и $l(w_i) = 2i + 1, 0 \leqslant i \leqslant N$, такую, что $w = w_K$ и $w_K\beta w_{K+1}\beta \ldots \beta w_N$ описывает все вычисление машины M_0 на IT в соответствии с соглашениями, в общих чертах набросанными при доказательстве 3.1. На любой другой ленте A' ведет себя так же, как $\bar{A}$.

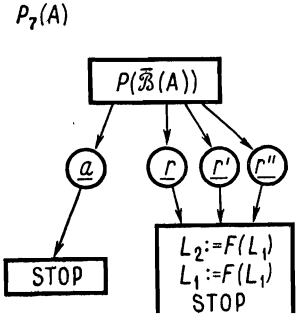
Теперь можно описать таблицу переходов для автомата $\bar{A}$. Она состоит из следующих правил:

$$\begin{array}{llll} q_0^1, & \beta \to \underline{r}, & q_0^1, & s_i \to q_1, \\ q_1^1, & \beta \to q_2, & q_1^1, & s_i \to \underline{r}, \\ q_2^1, & \beta \to \underline{r}, & q_2^1, & s_i \to q_3, \\ q_3^2, & \beta \to q_0, & q_3^2, & s_i \to q_2, \end{array} \qquad 2 \leqslant i \leqslant n.$$

Автомат A' будет сразу же описан, как только мы сопоставим его работу с работой автомата A, используемого при доказательстве 3.1. Сначала головкой 1 автомат A' проверяет, имеет ли его лента начальный сегмент $w_0\beta w_1\beta \ldots w_{K-1}\beta$, удовлетворяющий выше описанным условиям; если не имеет, то A' переходит в состояние $\underline{r}$. Тем временем головка 2 идет наравне с головкой 1, и если эта начальная проверка закончилась успешно, то обе головки находятся над началом w_K. Начиная с этой позиции, автомат A' моделирует вычисление, выполняемое автоматом A (в 3.1), до тех пор, пока не произойдет переход в отвергнутое состояние $\underline{r}$. Автомат A' переходит в состояние $\underline{r}$, если где-либо встречается символ β, не являющийся ограничителем полной конфигурации, или, наоборот, некоторая конфигурация не завершается символом β. С другой стороны, всякий раз, как A переходит в состояние $\underline{r}$, A' продолжает вычисление, но начиная с этого момента моделирует уже автомат $\bar{A}$. Так же как и в 3.1, существует конечное число (эффективно определяемое по IT) ситуаций, которые необходимо различать при работе автомата A', а поэтому A' может быть построен как конечный автомат.

Заметим, что, если лента отвергается как $\bar{A}$, так и A', головка 1 останавливается в обоих случаях в одной и той же точке ленты; последнее имеет место и для $\overline{\mathscr{B}}(\bar{A})$ и $\overline{\mathscr{B}}(A')$.

Пусть теперь $P_7(A)$ осуществляет преобразование автоматов A в схемы следующего вида:



Пусть $\bar{Y} = P_7(\bar{A})$. Тогда из выше описанного построения M_0 расходится на $IT \Rightarrow \mathcal{J}_a(\bar{\mathcal{B}}(A')) = \varnothing \Rightarrow P_7(A') \equiv \bar{Y}$.

С другой стороны, если M_0 останавливается на IT, то A' принимает некоторую такую ленту t, которая отвергается автоматом $\bar{A}$; например, пусть t — лента, которая отступает от формата, предписанного правилом (i), и имеет начальный сегмент, описывающий вычисление на IT; но тогда, выбрав I так, чтобы $t_I(F(L_1)) = t$, мы получим, что оба значения $\mathrm{Val}(P_7(A')_I)$ и $\mathrm{Val}(\bar{Y}_I)$ определены (и различны), а поэтому $P_7(A') \not\simeq \bar{Y}$. Следовательно,

$$M_0 \text{ останавливается на } IT \Rightarrow P_7(A') \not\simeq \bar{Y}.$$

Объединяя последние два результата, мы получим, что для всякого разумного отношения $\sim$

$$M_0 \text{ расходится на } IT \Leftrightarrow P_7(A') \sim \bar{Y};$$

к этому приходим так же, как и в случае 5.1. Теорема 5.2 доказана.

Этот раздел мы закончим результатом, который подводит итог следствиям теорем 5.1 и 5.2 в той их части, которая касается упрощения программы. Очевидно, что в силу 5.1 и 5.2 исключается существование любого алгоритма упрощения, позволяющего получить каноническую форму схем при таком разумном отношении $\sim$, которое является в строгом смысле отношением эквивалентности [1]); та же ситуация сохранится и при не слишком строгих ограничениях на отношение $\sim$, лишь бы они позволили такому алгоритму быть включенным в процедуру частичного разрешения отношения $P \sim Q$. С другой стороны, достаточно предположить существование некоторого алгоритма «исчерпывающего» упрощения, чтобы обеспечить получение такой канонической формы путем сведения схемы к «наипростейшей» ей эквивалентной. Ниже мы изложим те предпосылки относительно $\sim$ и метода упрощения, которые позволяют применить это рассуждение. Не всякое разумное отношение $\sim$ будет удовлетворять этим предпосылкам. Например, алгоритм, сопоставляющий всякой схеме схему D, которая состоит из одной инструкции и не останавливается ни на каких интерпретациях, при подходящем понимании «упрощения» является алгоритмом исчерпывающего упрощения относительно $\simeq$; таким образом, отношение $\simeq$ не будет удовлетворять предпосылкам, которые будут даны. Отметим, что из того, что может быть включено в понятие «упрощать», нами взято как можно меньше.

---

[1]) Напомним, что в общем случае отношение $\sim$ не обязано быть симметричным или транзитивным, т. е. может не быть отношением эквивалентности в строгом смысле. — *Прим. перев.*

Под методом упрощения для $\sim$ мы будем понимать эффективный метод, который по заданной схеме P перечисляет бесконечную последовательность P', P'', ... схем — «упрощений» схемы P (не обязательно различных), такую, что при всех n $P \sim P^{(n)}$. Исчерпывающим назовем такой метод, что всякий раз, как $P \equiv Q$, существуют такие m и n, что $P^{(m)} = Q^{(n)}$. В частности, если некоторый метод всегда приводит к «наипростейшей» схеме, если $\sim$ есть отношение эквивалентности и если $P \sim Q$ является (частично) разрешимым для «наипростейших» схем, то этот метод может быть расширен до исчерпывающего в том понимании этого слова, как это было дано выше. Подобные условия можно сформулировать для тех случаев, в которых существование «наипростейших» представлений схем является необязательным.

Теорема 5.3. *Если* $\sim$ *— разумное отношение и*

$$P \sim R \,\&\, Q \sim R \Rightarrow P \simeq Q, \qquad (*)$$

то для отношения $\sim$ *не существует метода исчерпывающего упрощения.*

Доказательство. Предположим, что такой метод существует, и придем к противоречию с 5.2 (или с 5.1). Заметим, во-первых, что для $P \in \mathscr{P}$ имеет место $\bar{Y} \sim P \Leftrightarrow \bar{Y} \equiv P \Leftrightarrow \bar{Y} \simeq P$. Рассмотрим частичную процедуру одновременного перечисления последовательностей P', P'', ... и $\bar{Y}'$, $\bar{Y}''$, ..., сопровождаемого проверкой на наличие в них общего элемента. Если $\bar{Y} \sim P$, то $\bar{Y} \equiv P$, таким образом, предположение об исчерпываемости ведет к тому, что для такой схемы P процесс успешно завершается; с другой стороны, если процесс для P завершается, то из (*) следует, что $\bar{Y} \simeq P$, и тогда $\bar{Y} \sim P$. Следовательно, $\bar{Y} \sim P$ в том и только том случае, если описанный процесс завершается для P, что противоречит 5.2. Этим теорема 5.3 доказана.

Заметим, что предпосылка (*) выполняется для любого разумного отношения $\sim$, более строгого, чем отношение «$<_F$», где $P <_F Q$, если $P \simeq Q$ и если Q останавливается на всякой конечной интерпретации, на которой останавливается P; и (*) не имеет места для отношения $\simeq$ или для отношения $>_F$, являющегося инверсией отношения $<_F$ (хотя применительно к этим последним понятиям упрощение не заканчивается особенно желательным образом).

6. МОНАДИЧЕСКИЕ СХЕМЫ ПРОГРАММ

В части, касающейся развития практической техники для оптимизации машинных программ и построения формальной системы для доказательства утверждений о них, главные резуль-

таты предыдущих разделов могут рассматриваться только как негативные. В этом разделе мы предложим результаты позитивного характера.

Сначала мы рассмотрим связь между схемами программ и другими абстрактными моделями вычислительных процессов. Схемы, все операторы которых используют одноместные символы, будут называться монадическими. Особенно простой подкласс монадических схем получается при том дополнительном ограничении, что все операторы присваивания обладают следующим свойством: ячейка присваивания совпадает с ячейкой обращения [1]) (т. е. $L_j := F_i(L_j)$); такие схемы называются схемами разобщенной памяти. Этот класс схем оказался сильно связанным с многоленточными автоматами Рабина — Скотта (см. [6]). Мы покажем, что не только взаимозаменяемы проблемы эквивалентности для многоленточных автоматов и для схем разобщенной памяти (т. е. одна проблема может быть сведена к другой), но что существует алгоритм для построения схемы, которая моделирует поведение данного автомата, и наоборот. Таким образом, можно говорить о том, что многоленточные автоматы являются моделями тех машинных программ, в которых результат всякого вычисления на одном наборе данных никогда не используется при любом [2]) вычислении на другом наборе данных.

Покажем, что многоленточные автоматы Рабина — Скотта могут рассматриваться как подкласс $\mathscr{RS}$ многоленточных автоматов, определенных в разделе 3. Алфавиты автоматов из $\mathscr{RS}$ содержат символ ε — «маркер конца», и эти автоматы обладают свойствами:

(i) Вычисление, выполняемое автоматом $A \in \mathscr{RS}$, завершается в том и только том случае, если каждая лента содержит вхождение символа ε.

(ii) Всякое такое вычисление завершается в ситуации, когда каждая головка находится непосредственно за первым вхождением символа ε на ее ленте.

Тогда класс $\mathscr{RS}$ «представляет» автоматы Рабина — Скотта, определенные в [6], в следующем смысле: существует эффективное соответствие между двумя классами, обладающее свойством: автомат Рабина — Скотта принимает или отвергает набор из n $\langle \hat{t}_1, \hat{t}_2, \ldots, \hat{t}_n \rangle$ конечных лент в том и только том случае, если соответствующий ему автомат из $\mathscr{RS}$ принимает или отвергает некоторый набор из n $\langle t_1, t_2, \ldots, t_n \rangle$ бесконечных лент, такой, что каждая t_i имеет начальным сегментом $\hat{t}_i\varepsilon$.

---

[1]) Здесь ячейкой обращения называется ячейка, входящая под знак функционального символа, а ячейкой присваивания — единственная ячейка, не входящая под этот знак. — *Прим. перев.*

[2]) Ведущемся параллельно с первым. — *Прим. перев.*

Определение: Два автомата A и B *эквивалентны* (записываем это в виде $A \equiv B$), если $\mathscr{T}_a(A) = \mathscr{T}_a(B)$ и $\mathscr{T}_r(A) = \mathscr{T}_r(B)$.

В частности, при $A, B \in \mathscr{RP}$ $A \equiv B$ в том и только том случае, если $\mathscr{T}_a(A) = \mathscr{T}_a(B)$, так как $\mathscr{T}_r(A) = X - \mathscr{T}_a(A)$, где X — это множество всех таких наборов n лент над Σ, каждая компонента которых имеет вхождение маркера конца.

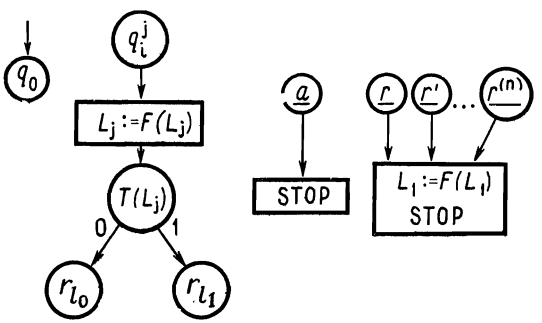
Во-первых, чтобы свести проблему эквивалентности для автоматов из $\mathscr{RP}$ к проблеме (строгой) эквивалентности для схем разобщенной памяти, покажем, как по n-ленточному автомату $A \in \mathscr{RP}$ сконструировать схему $\bar{P}$ с n разобщенными ячейками. Эта конструкция исходит из соображений, схожих с теми, что приведены в разд. 3 и 4.

Для $A \in \mathscr{RP}$ рассмотрим $\bar{\mathscr{B}}(A)$, где $\bar{\mathscr{B}}$ — преобразование из леммы 3.2; заметим, что при любых A, A'

$$A \equiv A' \Leftrightarrow \bar{\mathscr{B}}(A) \equiv \bar{\mathscr{B}}(A') \Leftrightarrow \mathscr{T}_a(\bar{\mathscr{B}}(A)) = \mathscr{T}_a(\bar{\mathscr{B}}(A'))$$

по свойствам (i), (ii), данным выше ((i) обеспечивает, что $A \equiv A' \Leftrightarrow \mathscr{T}_a(A) = \mathscr{T}_a(A')$; (ii) обеспечивает, что $\bar{\mathscr{B}}(A)$ и $\bar{\mathscr{B}}(A')$ достигают состояний $\underline{r}'$ и $\underline{r}''$ на множестве лент, в точности том же самом, на котором эти состояния являются мертвыми состояниями, введенными отображением $\bar{\mathscr{B}}$).

Пусть аналогично построению схемы $P(B)$ в теореме 4.1 схема $\bar{P}(A)$ получена по таблице бинарного автомата $\bar{\mathscr{B}}(A)$ путем отождествления меток в следующих неполных схемах:



Пусть теперь $t = \langle t_1, t_2, \ldots, t_n \rangle$ — некоторая бинарная n-лента и I — такая свободная интерпретация, что $t_I(L_i) = t_i$, $1 \leqslant i \leqslant n$. Очевидно, что для любой заданной бинарной t существует интерпретация I, связанная с t описанным образом, и наоборот. Пусть m_i — длина самого короткого начального сегмента ленты t_i, имеющего вид $\mathscr{B}(w\varepsilon)$ для некоторого слова w. Предполагая, что каждое m_i определено, обозначим через

$V(t)$ n-ку $\langle F^{m_1+1}L_1, F^{m_2+1}L_2, \ldots. F^{m_n+1}L_n\rangle$ выражений. Из построения схемы $\tilde{P}(A)$ мы имеем следующую лемму:

Лемма 6.1.

(i) *$\bar{\mathfrak{B}}(A)$ принимает t в том и только том случае, если $\tilde{P}_I(A)$ останавливается и* $\mathrm{Val}(\tilde{P}_I(A)) = V(t)$.

(ii) *Если $\bar{\mathfrak{B}}(A)$ и $\bar{\mathfrak{B}}(A')$ отвергают t, то $\tilde{P}_I(A)$ останавливается и* $\mathrm{Val}(\tilde{P}_I(A)) = \mathrm{Val}(\tilde{P}_I(A'))$.

(iii) *Если $\bar{\mathfrak{B}}(A)$ расходится на t, то $\tilde{P}_I(A)$ расходится.*

Из 6.1 и замечания, сделанного выше относительно отображения $\bar{\mathfrak{B}}$, когда оно ограничено классом $\mathfrak{RP}$, вытекает

Лемма 6.2. *Для любых A, $A' \in \mathfrak{RP}$ $A \equiv A'$ в том и только том случае, если $\tilde{P}(A) \equiv \tilde{P}(A')$. (Более строго, $A \equiv A' \Leftrightarrow \tilde{P}(A) \sim \tilde{P}(A')$ при любом разумном $\sim$, так как $\tilde{P}(A) \equiv \tilde{P}(A') \Leftrightarrow \tilde{P}(A) \simeq \tilde{P}(A')$.)*

Обратная проблема состоит в описании эффективного и сохраняющего эквивалентность преобразования $\mathfrak{A}$ схем разобщенной памяти в автоматы из $\mathfrak{RP}$. Пусть P — схема разобщенной памяти, включающая в себя, самое большее, символы ячеек $L_1, L_2, \ldots, L_n$, монодические функциональные символы $F_1, F_2, \ldots, F_m$ и, удобства ради, только один предикатный символ T (описанное ниже построение может быть очевидным образом модифицировано так, чтобы охватить схемы, содержащие более одного предикатного символа). Предположим, что I — некоторая свободная интерпретация и что $\mathrm{Val}(P_I) = \langle E_1, E_2, \ldots, E_n\rangle$; тогда $\mathfrak{A}(P)$ должен быть таким n-ленточным автоматом над алфавитом $\Sigma = \{0, 1, L_1, L_2, \ldots, L_n, F_1, F_2, \ldots, F_m, \varepsilon\}$, который принимает любую n-ленту $t = \langle t_1, t_2, \ldots, t_n\rangle$, удовлетворяющую некоторому условию, определяемому только интерпретацией I и выражениями $E_1, E_2, \ldots, E_n$ и не зависящему от P. Это условие состоит в том, что каждая лента t_i должна иметь начальный сегмент вида $w(I, E_i)\varepsilon$, где $w(I, E)$ для любого выражения E определяется следующим образом: если E — выражение

$$F_{I_k}F_{I_{k-1}}\ldots F_{I_1}L_{I_0}$$

то $w(I, E)$ — слово

$$L_{I_0}\delta_0 F_{I_1}\delta_1 \ldots F_{I_k}\delta_k,$$

такое, что $\delta_i = T_I(F_{I_i}F_{I_{i-1}}\ldots F_{I_1}L_{I_0})$. Существенные свойства, которыми должен обладать автомат $\mathfrak{A}(P)$, подытожены следующей леммой.

Лемма 6.3. *$\mathfrak{A}(P)$ принимает $\langle t_1, t_2, \ldots, t_n\rangle$ в том и только том случае, если каждая лента t_i имеет начальный сегмент вида*

$w(I, E_i)\varepsilon$, где I — некоторая свободная интерпретация, P_I останавливается и $\mathrm{Val}(P_I) = \langle E_1, E_2, \ldots, E_n\rangle$.

Построение $\mathfrak{A}$ осложняется технической трудностью; $\mathfrak{A}(P)$ должен быть построен так, чтобы сдвигать каждую ленту непосредственно после чтения с нее символа; поэтому, если символ на ленте должен регулировать более чем один переход из состояния в состояние, то должны быть предприняты специальные шаги, чтобы переписать этот символ во внутреннее состояние автомата. При построении $\mathfrak{A}(P)$, моделирующем P, эта трудность встречается всякий раз на символах, фиксирующих значения функции ветвления T_I. Чтобы преодолеть ее, схема P прежде всего должна быть преобразована в эквивалентную ей и более ограниченную схему, в которой ячейки проверяются только непосредственно после акта присваивания (фактически это и есть по Патерсону (см. [5]) процесс превращения «либеральной» схемы в «свободную»), либо автомат $\mathfrak{A}(P)$ прежде всего должен быть сделан как автомат с некоторой несущественной дополнительной способностью к переписи предварительно прочитанных символов, либо нужно задавать сразу прямое преобразование. Мы приняли второй подход. Мы прежде всего опишем автомат $\mathfrak{A}^*(P)$, который будет автоматом в смысле, определенном в разд. 3, но с добавкой некоторых «звездочковых» состояний, преобразования которых подчиняются особым правилам. Эти состояния помечаются ниже приписыванием надстрочного символа «*» во входах таблицы переходов. Преобразования состояния, обозначенного, например, через q^{i*}, зависят от предыдущей буквы, прочитанной с ленты i; в процессе такого преобразования ни одна лента не сдвигается; если же с ленты i еще не была считана ни одна буква, то за «предыдущую» букву принимается, например, символ «0». Детали автомата $\mathfrak{A}^*(P)$ выглядят следующим образом:

(i) Чтобы гарантировать принадлежность автомата $\mathfrak{A}(P)$ классу $\mathscr{RP}$, каждая лента прочитывается до ее маркера конца, прежде чем автомат $\mathfrak{A}^*(P)$ останавливается. Это осуществляют следующие $3n$ правил:

$$
\left.\begin{array}{llll}
r_1^{1*}, \varepsilon \to r_1, & r_1^{1*}, \xi \to \tilde{r}_1, \\
\tilde{r}_1^{1}, \varepsilon \to r_2, & \tilde{r}_1^{1}, \xi \to \tilde{r}_1, \\
\cdot & \cdot \\
\cdot & \cdot \\
\cdot & \cdot \\
\tilde{r}_{n-1}^{n-1}, \varepsilon \to r_n, & \tilde{r}_{n-1}^{n-1}, \xi \to \tilde{r}_{n-1}, \\
r_n^{n*}, \varepsilon \to \underline{r}, & r_n^{n*}, \xi \to \tilde{r}_n, \\
\tilde{r}_n^{n}, \varepsilon \to \underline{r}, & \tilde{r}_n^{n}, \xi \to \tilde{r}_n,
\end{array}\right\} \xi \neq \varepsilon.
$$

$$\left.\begin{array}{l} a_1^1,\ \varepsilon \to a_2, \quad a_1^1,\ \xi \to \tilde{r}_1, \\ \cdots \\ a_{n-1}^{n-1},\ \varepsilon \to a_n, \quad a_{n-1}^{n-1},\ \xi \to \tilde{r}_{n-1}, \\ a_n^n,\ \varepsilon \to \underline{a}, \quad a_n^n,\ \xi \to \tilde{r}_n, \end{array}\right\} \xi \neq \varepsilon.$$

Все переходы, ниже не описываемые, ведут к состоянию r_1.

(ii) По операторам $S_1, S_2, \ldots, S_p$ схемы P в автомате $\mathfrak{A}^*(P)$ вводятся соответствующие им состояния $s_1, s_2, \ldots, s_p$, определяемые следующим образом:

(a) Если S_l — это оператор STOP, то s_l — это состояние

$$a_1\ b\,(i).$$

(b) Если S_l — это оператор перехода

$$j.\ T\,(L_i)\,u,\ v,$$

то S_l имеет переходы

$$s_j^{i*},\ 0 \to s_u, \quad s_j^{i*},\ 1 \to s_v.$$

(c) Если S_l — это присваивание

$$j.\ L_i{:}\ = F_k\,(L_i),$$

то соответствующие правила имеют вид

$$\begin{array}{l} s_j^i,\ F_k \to \tilde{s}_j, \\ \tilde{s}_j^i,\ \delta \to s_{j+1}, \qquad \delta = 0,\ 1. \end{array}$$

(iii) $\mathfrak{A}^*(P)$ начинает работать в состоянии t_1, следуя правилам

$$\left.\begin{array}{l} t_1^1,\ L_1 \to \tilde{t}_1, \\ \tilde{t}_1^1,\ \delta \to t_2, \\ \cdots \\ \tilde{t}_{n-1}^{n-1},\ \delta \to t_n, \\ t_n^n,\ L_n \to \tilde{t}_n, \\ \tilde{t}_n^n,\ \delta \to s_1, \end{array}\right\} \delta = 0,\ 1.$$

Таким образом, $\mathfrak{A}^*(P)$ определяется как «звездочковый» автомат с $5n + 2m + l$ состояниями, где l и m соответственно число операторов перехода и присваивания в схеме P.

$\mathfrak{A}^*(P)$ «моделирует» схему P в очевидном смысле: предположим, что в некоторый момент вычисления P_I ячейки $L_1, L_2, \ldots, L_n$ «хранят» $E_1', E_2', \ldots, E_n'$ соответственно и должен быть выполнен оператор j; тогда, если каждая лента t_i имеет начальный сегмент $w(I, E_i')$, то в вычислении, выполняемом автоматом $\mathfrak{A}^*(P)$ на $\langle t_1, t_2, \ldots, t_n\rangle$, рассматриваемый момент достигается, когда головки лежат как раз за этими начальными сегментами и автомат $\mathfrak{A}^*(P)$ должен перейти в состояние s_j. Наоборот, при любом вычислении автомата $\mathfrak{A}^*(P)$ всякий раз, как достигается s_j, головки лежат как раз за начальными сегментами вида $w(I, E_i')$, такого, что существует момент в вычислении P_I, когда ячейки «хранят» $E_1', E_2', \ldots, E_n'$ и должен быть выполнен оператор j. Эти два свойства автомата $\mathfrak{A}^*(P)$ устанавливаются индукцией по длине неполных вычислений, осуществляемых соответственно схемой P и автоматом $\mathfrak{A}^*(P)$; 6.3 выполняется для $\mathfrak{A}^*(P)$, если рассматривать только такие j, которые являются адресами операторов STOP в P.

Теперь достаточно свести $\mathfrak{A}^*(P)$ и $\mathfrak{A}(P) \in \mathscr{RP}$, принимающему те же самые ленты. Это осуществляется следующим образом. За состояния автомата $\mathfrak{A}(P)$ принимаются $\underline{a}$, $\underline{r}$ и состояния $q_{q,\underline{x}}$, индексированные беззвездочковыми живыми состояниями автомата $\mathfrak{A}^*(P)$ и наборами из n $\underline{x} \in \Sigma^n$. Пусть функция $f_{\underline{x}}(q)$ для произвольного состояния q автомата $\mathfrak{A}^*(P)$ и произвольного набора из n $\underline{x} \in \Sigma^n$ определяется равенством

$$f_{\underline{x}}(q) = \begin{cases} p, & \text{если } q \text{ — звездочковое, где } p \text{ берется} \\ & \text{из правила } q^{i*},\ \underline{x}(i) \to p; \\ q & \text{в противном случае.} \end{cases}$$

Пусть N — число звездочковых состояний в $\mathfrak{A}^*(P)$. Заметим, что если $f_{\underline{x}}^N(q)$ — беззвездочковое, то это первое беззвездочковое состояние, встречающееся в последовательности переходов, которая начинается с q в ситуации, когда последняя буква, прочитанная каждой головкой i, есть $\underline{x}(i)$, и что в противном случае для той же исходной ситуации автомат $\mathfrak{A}^*(P)$ входит в «пустой цикл» (т. е. расходится, не передвигая ни одну из своих лент). Пусть $g(q, \underline{x})$ — следующая функция, значениями которой являются состояния автомата $\mathfrak{A}(P)$:

$$g(q, \underline{x}) = \begin{cases} \underline{a}, & \text{если } q' = \underline{a}, \\ \underline{r}, & \text{если } q' = \underline{r}, \\ q_{q',\underline{x}}, & \text{если } q' \text{ беззвездочковое}, \\ q_{f_{\underline{x}}(r_1),\underline{x}} & \text{в остальных случаях}, \end{cases}$$

где $q' = f_{\underline{x}}^{N}(q)$ и r_1 — состояние, введенное выше в (i) (заметим, что $f_{\underline{x}}(r_1)$ — беззвездочковое). Предположим теперь, что типичное беззвездочковое правило автомата $\mathfrak{A}^*(P)$ имеет вид

$$q^i,\ \xi \to p;$$

тогда для каждого $\underline{x} \in \Sigma^n$ соответствующее правило автомата $\mathfrak{A}(P)$ будет иметь вид

$$q^i_{q,\,\underline{x}},\ \xi \to g(p,\ \underline{y}),$$

где $\underline{y}$ получается из $\underline{x}$ заменой компоненты $\underline{x}(i)$ буквой ξ. Это завершает построение автомата $\mathfrak{A}(P)$. Заметим, что на любой n-ленте последовательность состояний, принимаемых автоматом $\mathfrak{A}(P)$, соответствует последовательности беззвездочковых состояний, принимаемых автоматом $\mathfrak{A}^*(P)$, если пренебречь компонентами $\underline{x}$ состояния автомата $\mathfrak{A}(P)$, с возможным исключением таких n-лент, которые $\mathfrak{A}(P)$ отвергает и на которых $\mathfrak{A}^*(P)$ расходится. Следовательно, $\underline{a}$ достигается автоматом $\mathfrak{A}(P)$ в том и только том случае, если $\underline{a}$ достигается автоматом $\mathfrak{A}^*(P)$ на той же самой n-ленте; таким образом, оба автомата принимают одни и те же ленты. Это завершает доказательство леммы 6.3.

Поскольку $\mathfrak{A}(P) \in \mathscr{RP}$, $\mathfrak{A}(P) \in \mathfrak{A}(Q)$ в том и только том случае, если $\mathscr{T}_a(\mathfrak{A}(P)) = \mathscr{T}_a(\mathfrak{A}(Q))$; прямым следствием леммы 6.3 является следующая лемма:

Лемма 6.4. *$P \equiv Q$ в том и только том случае, если $\mathfrak{A}(P) \equiv \mathfrak{A}(Q)$.*

Если P содержит r предикатных символов, то описанное выше построение может быть модифицировано требованием, чтобы слова $w(I, E)$ после каждого функционального символа содержали бинарные последовательности длины r; j-й член такой последовательности на ленте i интерпретируется как задающий «новое» значение j-му предикатному символу на L_i.

Можно до некоторой степени свободно говорить о двух вычислительных моделях как об эквивалентных, если проблемы эквивалентности в этих моделях являются взаимозаменяемыми и если существует алгоритм для получения по заданному элементу одной модели элемента другой модели, «копирующего» первый в некотором подходящем смысле. Факты, установленные выше, могут быть подытожены теоремой.

Теорема 6.5. *n-ленточные автоматы Рабина — Скотта эквивалентны схемам разобщенной памяти с n ячейками.*

Следствие. *Схемы Янова эквивалентны монадическим схемам с одной ячейкой.*

Следствие можно доказать непосредственно построением, подобным проведенному выше, или обращением к факту эквивалентности схем Янова конечным автоматам, установленному Ратледжем в [8].

В настоящее время неизвестно, является ли разрешимой проблема строгой эквивалентности для схем разобщенной памяти с n ячейками ($n > 1$); соответствующая проблема для n-ленточных автоматов является также открытой!

Ниже мы опишем несколько классов схемы, для которых известна процедура разрешения строгой эквивалентности, и несколько таких, для которых эта проблема остается открытой. Детали доказательств нами опущены и могут быть найдены в работе Патерсона [5].

Теорема 6.6. *Проблема эквивалентности для монадических схем с непересекающимися циклами является разрешимой.*

Мы все еще не в состоянии исключить ограничение, выражающееся в монадичности операторов, но полагаем, что результат сохранит свою силу. Процедура разрешения, применяемая в теореме, выражает эквивалентность двух заданных схем в виде формулы аддитивной теории натуральных чисел, которая является разрешимой теорией.

Характерной чертой схем с неразрешимыми проблемами разрешения, рассматриваемыми нами, является то, что большая часть вычисляемых выражений позже перевычисляется, что являлось бы необычным и нежелательным в настоящей машинной программе. Поэтому мы обратимся к различным ограничениям, которые можно наложить на схемы, чтобы предотвратить это свойство повторяемости.

Определение. Схема P называется свободной, если любая последовательность в P является последовательностью выполнения.

Определение. Схема P называется *либеральной*, если в любой последовательности в P ни одно выражение не вычисляется более чем один раз.

Теорема 6.7. *Существует эффективное построение по всякой заданной либеральной схеме эквивалентной ей свободной схемы.*

Представляется вероятным, что проблема разрешения эквивалентности свободных схем является разрешимой, и, конечно, никакая из техник, пока что используемых нами для доказательства неразрешимости, здесь очевидным образом неприменима. Теорема 6.7 показывает, что в практических целях любая

либеральная схема может рассматриваться как свободная, хотя обратное не имеет места, как это демонстрируется следующей свободной схемой:

$$L_2: = F(L_1),$$
$$\text{a. } L_1: = F(L_1),$$
$$L_1: = F(L_1),$$
$$L_2: = F(L_2),$$
$$T(L_1)\, b,\ a,$$
$$\text{b. STOP.}$$

Различия между этими двумя классами схем выявляются в полной мере следующими теоремами:

Теорема 6.8. *Свойство схемы быть свободной — неразрешимое.*

Теорема 6.9. *Свойство схемы быть либеральной — разрешимое.*

Доказательство теоремы 6.8 включает в себе сведение к комбинаторной проблеме Поста. В случае теоремы 6.9 можно показать, что возможен эффективный и исчерпывающий поиск первого проявления нелиберальности.

Для начала мы располагаем процедурой разрешения только для подкласса либеральных схем.

Определение. Схема называется *прогрессивной*, если в любой ее последовательности ячейка присваивания каждого вычислительного оператора берется в качестве ячейки обращения в следующем вычислительном операторе, если таковой существует.

Теорема 6.10. *Проблема эквивалентности для прогрессивных схем является разрешимой.*

Процедура разрешения подобна той, что используется при разрешении эквивалентности конечных автоматов, но, кроме того, включает в себя симметрическую группу всех перестановок ячеек схемы.

ЛИТЕРАТУРА

1. Davis M., Computability and Unsolvability, McGraw-Hill Book Co., New York. 1958.
2. Янов Ю. И., Логические схемы алгоритмов, Проблемы кибернетики, 1 (1960), 82—140.
3. Lucknam D., Park D., The undecidability of the equivalence problem for program schemata, Bolt, Beranek, and Newman Inc., Report No. 1141, Santa ana, Calif., 1964.

4. McCartny J., A basis for a mathematical theory of computation, in Computer Programming and Formal Systems (P. Braffort and D. Hirschberg, Eds.), pp. 33—70, North-Holland, Amsterdam, 1963.
5. Paterson M., Equivalence Problems in a Model of Computation, Doctoral Dissertation, Cambridge University, 1967.
6. Rabin M., Scott D., Finite automata and their decision problems, *IBM J. Res. Develop.* 3 (1959), 114—125.
7. Rosenberg A., On multi-head finite automata, Proceedings of the fifth Annual Symposium on Switching Circuit Theory and Logical Design, pp. 221—228, 1963.
8. Rutledge J., On Ianov's program schemata, *J. Assoc. Comput. Mach.* 11 (1964), 1—9.

Символ: большая экспериментальная система для изучения возможности погружения программного обеспечения в аппаратуру[1])

Уильям Р. Смит, Рэкс Райс, Гилман Д. Чеслей, Теодор А. Лейлиотис, Стефан Ф. Ландстром, Майрон А. Кэлхаун, Лоуренс Д. Джераулд, Томас Г. Кук

Фаэрчайлд Кэмера энд Инстрамент Корпорейшн, Пало Алто, Калифорния

ВВЕДЕНИЕ

Система СИМВОЛ — это результат широких исследований, направленных на увеличение функциональных возможностей аппаратуры. Проект основан на общей ревизии принципов построения вычислительных систем и посвящается следующим вопросам: пересмотру традиционного разделения между аппаратурой и программным обеспечением, пересмотру роли программных команд и памяти данных и методам уменьшения общей сложности и стоимости вычислений [1]. Для того чтобы достаточно точно оценить реализацию принятых принципов, было решено построить реальную работоспособную экспериментальную систему. Система СИМВОЛ, которая в настоящее время уже эксплуатируется, является результатом этих усилий.

При разработке системы вопросы аппаратного и программного обеспечения не разделялись. В сущности никто из работавших над проектом не мог назвать себя чистым специалистом по аппаратуре или программному обеспечению. Например, логическая разработка процессора по работе с полями была сделана человеком с квалификацией хорошего программиста [2]. Автоматизация составления монтажных схем осуществлена инженером, который до этого был узким специалистом по разработке логики.

Многие практические вопросы разработки аппаратуры с большими возможностями были изучены еще до того, как система вступила в действие. Мы не беремся утверждать, что СИМВОЛ представляет собой оптимальную многопроцессорную систему общего назначения с разделением времени. Наоборот, многочисленные упрощения приняты в тех местах, которые не слу-

[1]) Smith W. R., and al., SYMBOL — A large experimental system exploring major hardware replacement of software, Spring Joint Computer Conf., 1971, p. 601—616.

жили основным целям проекта. Примером этого может служить некоторое ограничение модульности. Однако мы утверждаем, что СИМВОЛ представляет собой значительный шаг вперед в технике создания систем и закладывает основы для значительного уменьшения стоимости вычислений. Так как разработка системы вступает в такую фазу, когда ее можно оценить по-настоящему, нужно показать ее реальную ценность для исследования будущих систем.

В данной статье представлено схематическое описание системы СИМВОЛ. Вместо того чтобы в общем виде рассматривать многие вопросы, мы попытаемся на упрощенных примерах объяснить основные свойства системы.

ОБЩАЯ ОРГАНИЗАЦИЯ

Система включает в себя восемь специализированных процессоров, каждый из которых работает как автономное устройство. Каждое функциональное устройство связано с системой при помощи главных шин (см. рис. 1). Некоторые свойства системы и их отношение к общей организации процессоров рассматриваются в общих чертах в следующих разделах.

Динамическое распределение памяти

Непосредственное распределение памяти аппаратурой, пожалуй, наиболее замечательная особенность системы СИМВОЛ. Распределением памяти занимается специализированный процессор, называемый Управлением Памяти (УП). УП целиком изолирует главную память от основных шин и других процессоров, и в нем предусмотрены достаточно сложные операции работы с памятью, которые он выполняет для других процессоров. Вместо обычных операций над памятью типа «читать/писать» в УП есть набор из пятнадцати операций, которые доступны другим процессорам системы. УП — это процессор специального назначения, который распределяет память по требованию, выполняет адресную арифметику и управляет ассоциативной памятью, с помощью которой осуществляется страничная организация. Другой процессор, Восстановление Памяти (ВП), дополняет УП, переорганизуя возвращенные области памяти, для того чтобы сделать их пригодными для нового использования. Он оформлен в виде отдельного устройства, работающего как задача с низким приоритетом доступа в память.

Прямая компиляция

Транслятор (ТР) воспринимает язык высокого уровня СИМВОЛ [3] в качестве входного и создает объектную строку обратной польской записи и таблицу имен, предназначаемые для обработки Центральным Процессором (ЦП). Используя неболь-

шую (порядка 100 слов) таблицу, находящуюся в главной памяти, ТР выполняет и прямую аппаратную компиляцию.

Динамическое изменение длины поля

Внутри Центрального Процессора все операции с полями выполняются как действия над полями с динамически переменной длиной. Обработка всех буквенно-цифровых строк выполняется Форматным Процессором (ФП), а обработка всех чисел производится Арифметическим Процессором (АП). При работе с данными ресурсы УП интенсивно используются ЦП.

Динамическое изменение структур данных

Разрешается любое изменение структур данных. Во время обработки они могут менять размер, формат и глубину. Ссы-

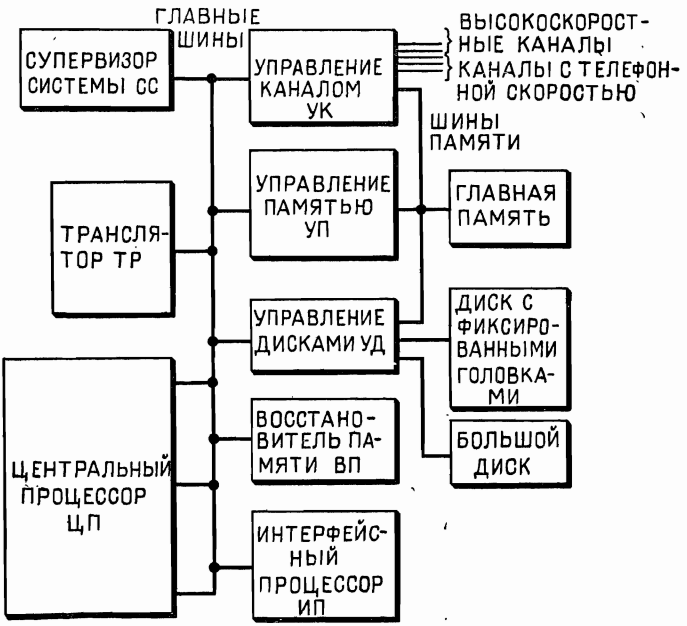


Рис. 1. Общая блок-схема системы СИМВОЛ.

лочный Процессор (СП), являющийся частью ЦП, управляет размещением и выборкой, хранением и ссылкой на массивы и структуры данных. СП интенсивно пользуется операциями УП.

Управление работой с разделением времени

Супервизор Системы (СС) составляет расписание задач для системы. Он управляет всеми переходами от одного режима обработки к другому и управляет очередями ко всем процессорам,

работающим с разделением времени. СС выполняет два важных аппаратных алгоритма — составляет расписание задач и управляет страничной организацией памяти. При нормировании дефицитных ресурсов, таких, как время Центрального Процессора, используются часы реального времени. СС также осуществляет ключевые передачи информации, необходимые для связи аппаратных алгоритмов с процедурами системы программного обеспечения.

Прямое редактирование текста

Ввод/вывод задач в системе осуществляется Интерфейсным Процессором (ИП) и Управлением Канала (УК). ИП имеет возможность производить общее редактирование текста при интерактивной связи через специальный терминал. Ввод/вывод и редактирование текста не используют ресурсы ЦП.

Управление виртуальной памятью

Когда УП обнаруживает, что нужная страница отсутствует в главной памяти, он оповещает об этом затребовавший страницу процессор и супервизор системы. Затем СС, используя страничный алгоритм, выдает соответствующие команды передачи с диска Управлению Дисками (УД). Каждое устройство, использующее память, после получения отказа, указывающего, что нужная страница отсутствует в главной памяти, должно уметь приостановить свою работу, запомнить текущее состояние и начать снова свою работу после того, как нужная страница будет вызвана в память.

КОНФИГУРАЦИЯ СИСТЕМЫ

В системе имеется небольшое дополнительное оборудование для периферии и памяти, связанное с основной стойкой. Этого дополнительного оборудования достаточно для экспериментальных целей системы. Главная память размера в 8К слов по 64 разряда в слове представляет собой ферритовую память с циклом в 2,5 мксек. Она разбита на 32 страницы по 256 слов в странице. Оперативная страничная память находится на малом диске с фиксированными головками фирмы Барроуз и разделена на 500 страниц. Массовая память страниц находится на большом диске фирмы Дейта Продактс и разбита на 50 000 страниц.

Управление Каналом разработано с таким расчетом, чтобы оно могло управлять 31 каналом. Считалось, что такое ограничение на количество каналов не помешает правильно оценить экспериментальную систему. В описываемом случае применены один высокоскоростной канал (с эффективным быстродействием 100 000 бит/сек) и три канала телефонных линий со скоростью до 2400 бод. В дальнейшем могут быть добавлены дополнительные каналы.

Основная стойка содержит около 18 000 сдвоенных нелинейных комбинированных транзисторных микросхем ($CT\mu L$). Их физические данные описаны в других статьях [4], [5]. Относительные размеры автономных процессоров показаны на рис. 2.

Рис. 2. Схема распределения аппаратуры СИМВОЛа, отражающая относительные размеры различных процессоров.

СВЯЗИ СИСТЕМЫ

Главные шины системы являются основным путем связи и могут использоваться в режиме разделения времени. При их разработке использовались специальные свойства комбинированных транзисторных логических микросхем. Шины включают в себя 111 параллельных линий, которые распределены следующим образом:

Шины данных	64
Шины адреса	24
Шины кода операции	6
Шины идентификации терминала	5
Шины приоритета	10
Часы системы	1
Сброс системы	1

Возможны следующие четыре типа использования шин:

Передачи от процессора к УП
Передачи от УП к процессору
Передачи от процессора к процессору
Циклы обмена управляющей информацией

Передача информации выполняется в соответствии с ее приоритетами. Шины приоритета указывают, для чего собираются использовать шины передачи в следующем цикле. Устройство, которое хочет воспользоваться шинами, возбуждает свою линию приоритета и проверяет, нет ли на шинах запроса с более высоким приоритетом, и если нет, то использует шины в следующем цикле.

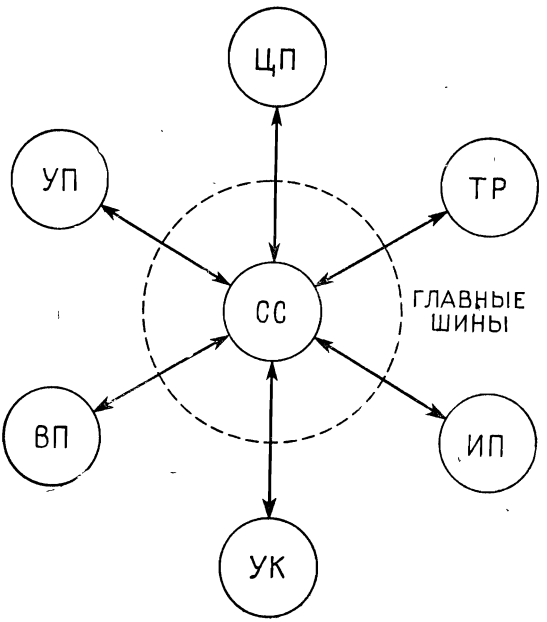


Рис. 3. Использование главных шин для циклов обмена управляющей информацией.

Циклы обмена управляющей информацией используются для обмена управляющей информацией между СС и другими процессорами через шины адреса и данных. См. рис. 3. Во время управляющего цикла линии шин адреса и данных используются заранее заданным образом. При этом определенные линии используются для запуска ЦП. Другие употребляются для указания режима завершения работы ТР. Во время данного цикла можно использовать любые комбинации путей. СС имеет такие свои собственные операции управления интерфейсом, которые он использует для связи с процессорами во время управляющего цикла, что более чем один сигнал может быть передан за время данного цикла.

ОРГАНИЗАЦИЯ ПАМЯТИ

Виртуальная память

Память системы СИМВОЛ организована как простая двухуровневая память с фиксированным размером страницы [6]. Страница состоит из 256 слов, каждое из которых имеет 64 разряда. Слово из виртуальной памяти выбирается по 24-разрядному адресу, 16 разрядов которого используются для выборки

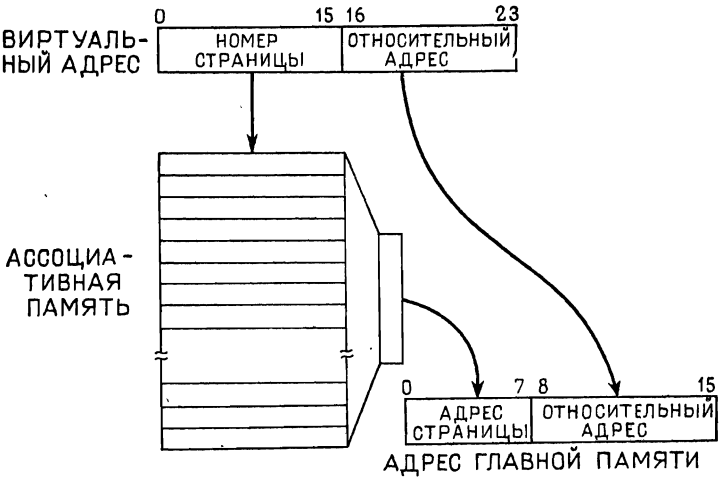


Рис. 4. Структура простой двухуровневой адресации виртуальной памяти.

страницы, а 8 разрядов — для выборки заданного слова внутри страницы (см. рис. 4).

Главная память экспериментальной системы разделена на 32 страницы. Относительная часть адреса (внутри страницы) используется непосредственно, в то время как адрес страницы нужен для обращения к ассоциативной памяти, из которой получается текущий адрес страницы в главной памяти.

В ассоциативной памяти для каждой страницы главной памяти отведена одна ячейка. Так как ассоциативная память связана с главной памятью, отдельные процессоры освобождаются от работы по определению местонахождения информации. Это ведет к значительному уменьшению логической сложности процессоров, хотя и может повести к некоторому увеличению количества электронных компонентов.

Положение страницы в страничной памяти на диске жестко закреплено (см. рис. 5). По требованию страница переписывается в свободное место главной памяти. При изгнании из памяти страница записывается в то же самое место на диске. Если

содержимое страницы не изменялось в главной памяти, то фактическая перепись в этом случае не производится.

Организация главной памяти показана на рис. 6. Первая страница отведена под таблицы системы, которые включают в себя таблицу слов, зарезервированных для транслятора, таблицу вызова программного обеспечения и управляющие слова для распределения памяти и постановки в очередь. Следующая группа страниц используется для записи управляющих слов различных терминалов или пользователей системы. Каждый активный терминал имеет 24 слова управляющей информации в этой небольшой памяти. Конечно, если система будет иметь большую мощность по каналам, то большее количество управляющей информации придется помещать в виртуальную память. Для простоты в СИМВОЛе введено следующее ограничение — все таблицы канала помещаются в главную память.

Буферы ввода/вывода различных активных каналов также находятся в ферритовой памяти. Для каждого активного канала требуется буфер размером в 16 слов. Буферы переменной длины не используются, хотя это и могло бы быть сделано.

Остальная часть главной памяти доступна как буфер для виртуальной памяти. Управление страницами осуществляется аппаратурой, причем выбором страницы, которая изгоняется из памяти, управляет Супервизор Системы. Этот алгоритм представляет собой очень гибкий параметризованный процесс, который позволяет выполнять большинство обычных страничных алгоритмов. Для каждого терминала имеются свои параметры, так что страничная динамика может изменяться при переходе от терминала к терминалу.

В противоположность более сложным схемам сегментации [7], [8] организация виртуальной памяти СИМВОЛа очень проста. Разница заключается в том, что в СИМВОЛе не нужна адресация уровня выше страничного. Это позволяет упростить схему виртуальной памяти в системе. Все пользователи и каналы делят между собой одну и ту же область виртуальной памяти, 24-разрядный адрес используется полностью. Так как память распределяется по требованию и нет ограничения на порядок присваивания страниц пользователю, то можно предсказать, что 24 разрядов в адресе будет достаточно для гораздо большего числа терминалов, чем 31. Если размер файла таков, что для его адресации нужно более 24 разрядов, то к его записям можно адресоваться при помощи специальных блочных передач ввода/вывода.

Списки страниц

Страницы объединены между собой в списки страниц. Свободные страницы организованы в отдельный список свободных

страниц. Если какому-либо пользователю требуется место в памяти, то открывается список страниц пользователя и из списка свободных страниц передается страница в список пользователя. В это же время создается управляющее слово, которое служит отправной точкой во всей дальнейшей работе пользователя со

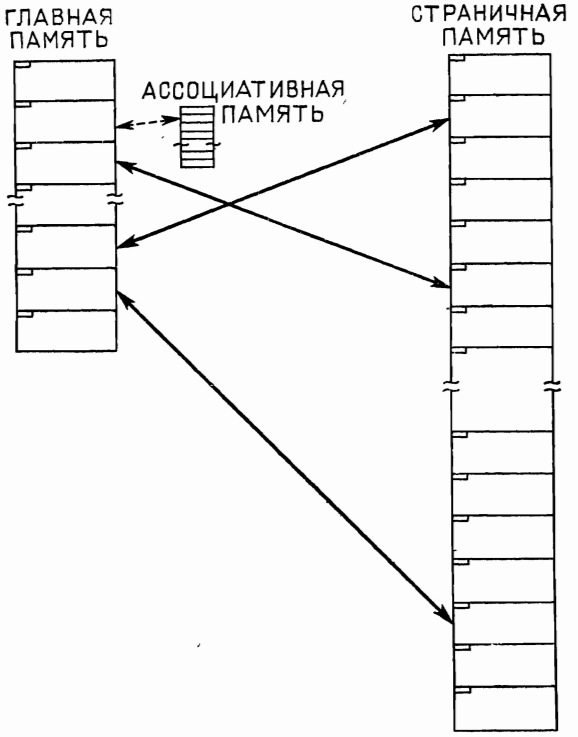


Рис. 5. Организация виртуальной памяти, показывающая фиксированное расположение страниц в страничной памяти.

списком страниц. Если потребуется дополнительное место в памяти, то добавляются новые страницы, образуя область памяти переменной длины, используемую для самых общих целей (см. рис. 7).

Пользователь может иметь более чем один список страниц. Типичным методом использования списков страниц для терминала является такой: один список страниц содержит текст исходной программы, другой — скомпилированную объектную программу, третий предназначен для памяти переменных данных. Другие списки страниц могут использоваться для долговременного или кратковременного хранения файлов. Списки страниц

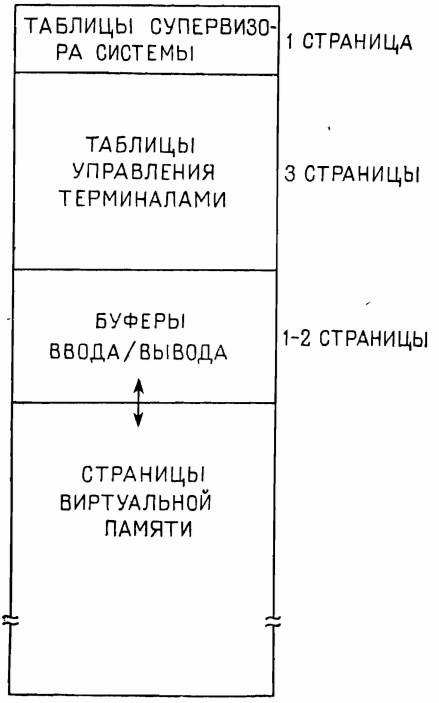


Рис. 6. Схема расположения информации в главной памяти.

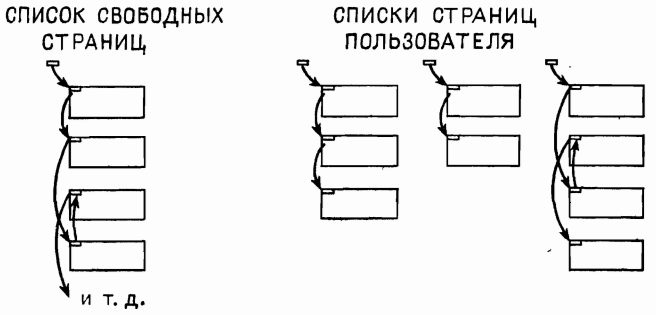


Рис. 7. Упрощенная структура списка страниц внутри виртуальной памяти.

постепенно растут по мере того, как требуется место в памяти. А когда список страниц целиком становится не нужен, он отдается назад системе, а та возвращает его в список свободных страниц.

Организация страницы

Для того чтобы легко оперировать с областями памяти, имеющими несмежные адреса, некоторое дополнительное количество памяти должно быть отведено для связывания или соединения отдельных частей. В СИМВОЛе около 11% памяти расходуется для постоянных записей, служащих для такой организации памяти.

Каждая страница имеет три отдельных информационных области, как показано на рис. 8. Первая область, называемая заголовком страницы, используется для организации списков страниц и управляет распределением памяти внутри страницы. Вторая область включает в себя 28 слов. Третья область состоит из 28 восьмисловных групп. Каждая группа имеет соответствующее слово групповой связи. Связь между группой и его словом определяется их положением в странице. Примером может служить слово 5 и соответствующая ему группа 5, заштрихованные на рис. 8. Данные пишутся в слова с 28 по 2F. Восьмисловная группа является неделимым квантом памяти. Это минимальный участок памяти, который может быть выдан при запросе.

Если для какой-то цели нужны данные, то память под них выдается целыми группами. Например, если нужно шесть слов памяти для записи вектора данных, то выдается одна группа. Если для записи вектора нужно 14 слов в памяти, то будет выделено две группы. Области информации различной длины образуются при помощи связывания между собой этих основных единиц памяти.

Строки информации

В общем случае для запоминания информации в СИМВОЛе используются списки переменной длины, состоящие из ячеек памяти. Это логически последовательные ячейки памяти, но не обязательно физически последовательные ячейки.

Рассмотрим типичную строку информации переменной длины, показанную на рис. 9. Память данных для 24 слов информации связана между собой при помощи своих групповых слов связи. Если начальный адрес строки известен, то можно выбрать всю строку, выбирая соответствующее групповое слово связи, каждый раз, как встречается конец группы. Можно пройти строку и в обратном направлении, используя ссылку назад, которая также записана в групповом слове связи.

Каждый процессор пользуется аппаратом переменной длины памяти, находящимся в Управлении Памяти (УП), не заботясь о сложной фактической последовательности адресов. Например, когда процессору нужно место в памяти для записи полей вектора данных, он посылает в УП команду Выдать Группу (ВГ)

Рис. 8. Организация страницы, показывающая группы и расположение слов связи. Адреса даны в шестнадцатеричной системе.

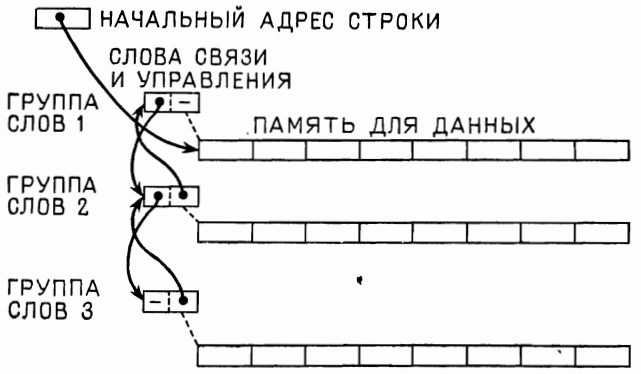


Рис. 9. Структура строки переменной длины.

вместе с меткой (tag), задающей тот список страниц, в который должна быть записана эта строка. УП затем выбирает свободную группу из списка страниц и возвращает адрес первого слова в группе процессору, давшему запрос. Когда процессор готов записать слово, он передает в УП данные, предварительно полученный адрес и команду Записать и Выдать (ЗВ). УП записывает данные и возвращает адрес следующего свободного слова. После того как встретился конец группы или строки, УП выделяет новую группу и привязывает ее к строке.

В процессе записи строки процессор получает адрес от УП и позднее сам передает этот адрес в УП для дальнейшего нара-

щивания строки. Вся адресная арифметика выполняется в УП. Рассмотрим пример, приведенный в табл. 1. В результате первых пяти команд слова A, B, C и D записываются в строку, начинающуюся со слова A.

Для того чтобы выбрать теперь записанную строку, в УП посылаются первоначальный адрес слова A и команда Выбрать и Продолжить (ВП). Данные из ячейки A возвращаются вместе со следующим логическим адресом. Когда строка больше не нужна, в УП выдается команда Освободить Строку (ОС) и начальный адрес этой строки. После этого строка целиком помещается в список для восстановления. Процессор Восстановления Памяти (ВП) сканирует списки для восстановления различных списков страниц в то время, когда память не занята другой работой и делает группы отданной строки доступными для нового распределения.

Основной процесс использования памяти имеет дело с разновидностями операций ВГ, ЗВ, ВП и ОС. Есть еще одиннадцать других команд памяти, которые вместе с вышеупомянутыми составляют полный комплект команд обслуживания памяти.

При проектировании памяти СИМВОЛа уделялось большое внимание эффективности использования места в памяти. Было

Таблица 1

Упрощенный пример последовательного использования памяти

Мнемоника	Операция	Адрес в УП	Возвращенный адрес	Данные в УП	Возвращенные данные
ВГ	Выдать Группу	—	a	—	—
ЗВ	Записать и Выдать	a	b	A	—
ЗВ	Записать и Выдать	b	c	B	—
ЗВ	Записать и Выдать	c	d	C	—
ЗВ	Записать и Выдать	d	c	D	—
ВП	Выбрать и Продолжить	a	b	—	A
ВП	Выбрать и Продолжить	b	c	—	B
ВП	Выбрать и Продолжить	c	d	—	C
ВП	Выбрать и Продолжить	d	—	—	D
ОС	Ссвободить Строку	a	—	—	—

проведено исследование по определению оптимального размера распределяемой группы [9]. Основные усилия были направлены на то, чтобы уравновесить стоимость лишней памяти, затрачиваемой на связи, и стоимость неиспользуемых кусков в группах. Дополнительные затраты компенсируются тем, что память распределяется по требованию. В большинстве машин выдается фиксированный кусок памяти, равный максимальному размеру массива данных. Когда используются массивы средних размеров, то, несмотря на значительный перерасход памяти на организацию распределения по требованию, этот метод может оказаться более выгодным, чем обычное распределение фиксированными кусками максимального размера.

ФОРМЫ ИНФОРМАЦИИ

Поля данных

В системе существует два основных типа данных — поля строк и числовые поля. Поле строки можно описать следующим образом. В начале строки стоит специальный символ Начала Строки (НС — SS). За ним следует произвольное количество алфавитно-цифровых символов в коде ASCII [1]). Заканчивается строка специальным символом Конца Строки (КС — SE). Это иллюстрирует, пожалуй, самую существенную черту представления данных в СИМВОЛе. Тип и длина данных содержатся вместе с самими данными. Код команды не зависит от динамических атрибутов данных.

Второй тип данных — это десятичное упакованное число с плавающей запятой переменной длины. Числовая форма имеет еще и указатель класса точности. Числа могут быть *точными* с бесконечным числом нулей в конце или они могут быть *эмпирическими*, что означает, что дальнейшие цифры неизвестны. Как и в поле строки, все атрибуты данных содержатся в памяти вместе с самими данными.

Для упрощения разработки аппаратуры другие формы представления данных не использовались. Предстоит еще включить в СИМВОЛ такие формы, как плотно упакованные двоичные строки переменной длины, двоичные числа фиксированной длины, двоичные числа переменной длины и т. д. В любом случае данные должны нести указатель типа и явное или неявное указание длины поля.

Исходные программы

Исходные программы — это поля строк специального вида. Это строки символов переменной длины в коде ASCII с разде-

[1]) ASCII (AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE) — Американский стандартный код для обмена информацией.

лителями, определяющими длину и тип. Во время подготовки их можно трактовать как поля данных, а позднее использовать как исходную программу для компиляции. Исходные программы могут быть собраны в библиотеки различных форматов, лишь бы они продолжали сохранять атрибуты поля строки для целей компиляции.

Структура данных

Структура данных определяется как группа переменной длины, состоящая из элементов, которые могут быть либо полями строки, либо числовыми полями, либо снова группами элементов. При таком рекурсивном определении структура может быть

⟨John | Alice | Jim | Elizabeth⟩

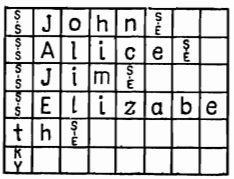


Рис. 10. Вектор полей строки и соответствующее представление данных в памяти.

вектором, матрицей или нерегулярным массивом. За исключением того, что поле или группа во время выполнения программы не должны превышать размера главной памяти, не накладывается никаких других ограничений на глубину или размер массива.

Для примера рассмотрим простой вектор, показанный на рис. 10. Введены специальные знаки $<$, | и $>$ для обозначения границ поля и группировки полей. Они называются соответственно — левый знак группы, знак поля и правый знак группы. В памяти системы поля строки ограничиваются символами Начала Строки (НС) и Конца Строки (КС). Еще одним специальным символом, называемым Концом Вектора (КВ), заканчивается группа полей. В памяти, представленной на рис. 10, показано несколько полей строк, после которых следует код Конца Вектора (КВ), который снова служит указателем длины, записанным вместе с данными. Поля строки располагаются с начальной границы машинного слова. В случае с Elizabeth, для того чтобы записать поле, требуется два машинных слова.

Представление матрицы на рис. 11 подобно представлению вектора, за исключением того, что существуют два уровня векторов. Теперь определение структуры можно дать следующим образом. Структура — это группа переменной длины, состоящая

из элементов, каждый из которых может быть либо полем строки, либо числовым полем, либо адресной ссылкой на другую группу.

Объектная строка и таблицы имен

При компиляции программы транслятор создает строку обратной польской записи и структурную таблицу имен. Польская

```
⟨⟨ 2 N 4 3 2 | P N P | . 1 7 ⟩
 ⟨ 2 N 7 0 8 A P C 1 4 3 | N P N | . 3 8 ⟩⟩
```

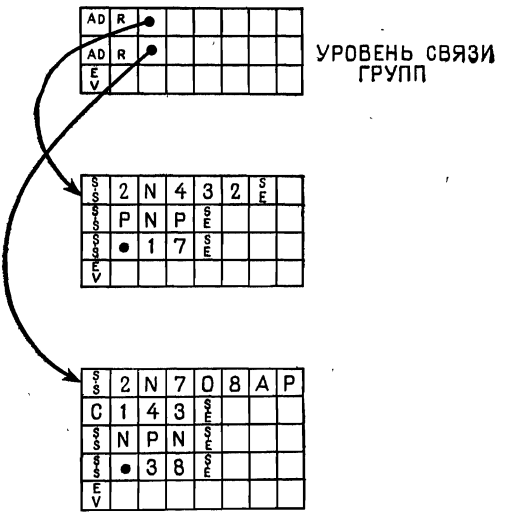


Рис. 11. Простой двумерный массив и соответствующие ему три строки переменной длины в памяти.

строка, называемая объектной строкой, и таблица имен являются основными формами информации, используемой Центральным Процессором во время исполнения программы. Употребление отдельных таблиц имен во время исполнения программы, возможно, является самым большим отступлением от традиционных форм обработки. Тогда как в большинстве систем программная строка для своего исполнения содержит адресные ссылки на место в памяти, где находятся данные, в системе СИМВОЛ объектная строка содержит ссылку на запись в таблице имен, которая служит централизованным местом, где содержится вся информация о данном идентификаторе. Это как раз та особенность системы, которая придает ей исключительный динамизм. Как бы ни менялся элемент с данным

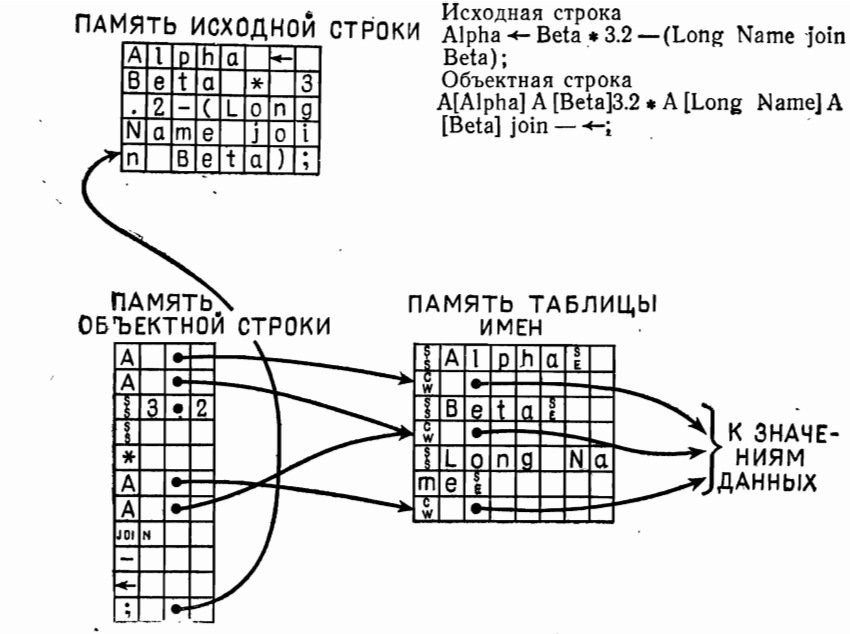


Рис. 12. Структура информации для простого оператора присваивания.

Значения данных
Alpha ⟨John Doe|110 Main ⟨30—25|DSR⟩ ⟨39|MS|12|⟩⟩
Beta|1432.1|
Gamma|Heading for a report|

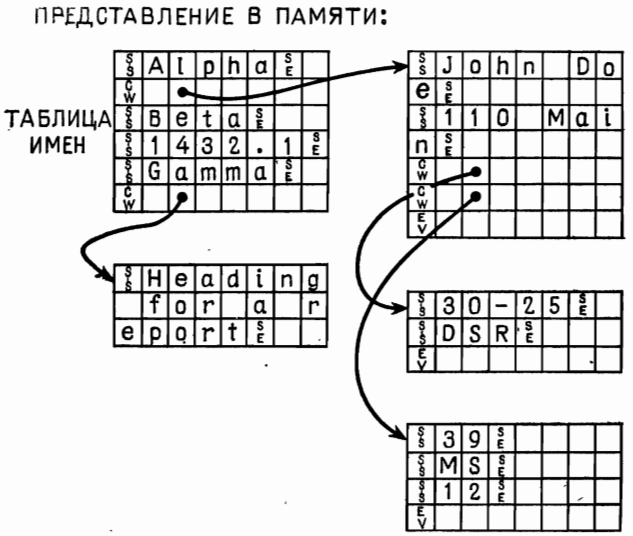


Рис. 13. Пример структуры и двух полей и способа их записи в памяти вместе с таблицей имен.

идентификатором: по месту расположения, по размеру, по типу и т. д., достаточно изменять в программе лишь запись в таблице имен, так как все ссылки из объектной строки на идентификатор должны идти через эту запись.

Исходная форма простого оператора присваивания и соответствующая ему объектная строка и таблица имен показаны на рис. 12. Идентификаторы выделены, и, если их до этого не было в таблице имен, то они туда добавлены. Заметим, что сами идентификаторы могут иметь переменную длину и состоять более чем из одного слова. С каждым идентификатором связано управляющее слово. Объектная строка состоит из адресов таблицы имен, литеральных данных (значение 3.2), операторов в обратной польской записи и соответствующих ссылок назад на исходную строку. Эти ссылки используются лишь при диагностике ошибок и игнорируются при нормальном исполнении программы. Объектная строка и таблица имен совершенно не зависят от будущего размера и типа данных переменной.

Теперь рассмотрим таблицу имен после того, как началось исполнение программы, и предположим, что переменные имеют текущие значения. На рис. 13 переменные Beta и Gamma являются простыми полями. Gamma представляет собой строку из нескольких слов, и поэтому она записана в строку памяти вместе с адресной ссылкой, помещенной в соответствующее управляющее слово. Beta является коротким полем и поэтому может быть помещена в одном слове прямо в таблице имен. Alpha представляет собой нерегулярную структуру. Для Alpha в таблице имен содержится ссылка на первую группу, которая в свою очередь содержит два поля строки, две адресные ссылки и знак конца вектора. Адресные ссылки указывают на две группы, одна из которых содержит два поля, а другая — три. В процессе исполнения программы атрибуты и представление в памяти переменных могут меняться. В любом случае таблица имен и сами данные будут содержать все атрибуты переменных.

ОСНОВНОЙ ПОТОК ИНФОРМАЦИИ

Для того чтобы понять, как различные процессоры системы СИМВОЛ участвуют в решении задачи конкретного пользователя, надо временно пренебречь тем, что система многопроцессорная. Пользователь за терминалом работает в различных режимах. Основными режимами являются загрузка программы, ее компиляция и исполнение. Рассмотрим диаграмму состояний на рис. 14. Пользователь начинает работать в режиме, когда терминал не управляется системой (режим АВТОНОМНО) и при помощи какого-то особого управления переводит свои задачи в холостой режим под управлением системы (холостой режим

ОПЕРАТИВНО). Отсюда он может перейти в режим ЗАГРУЗКА, для того чтобы создавать и дополнять свою программу. Когда пользователь готов выполнять программу, то в предположении, что он отличный программист, он начнет компилировать свою программу, а затем ее выполнять. По окончании выполнения программы программист может запустить ее заново или вернуться в режим ЗАГРУЗКА и изменить свою программу.

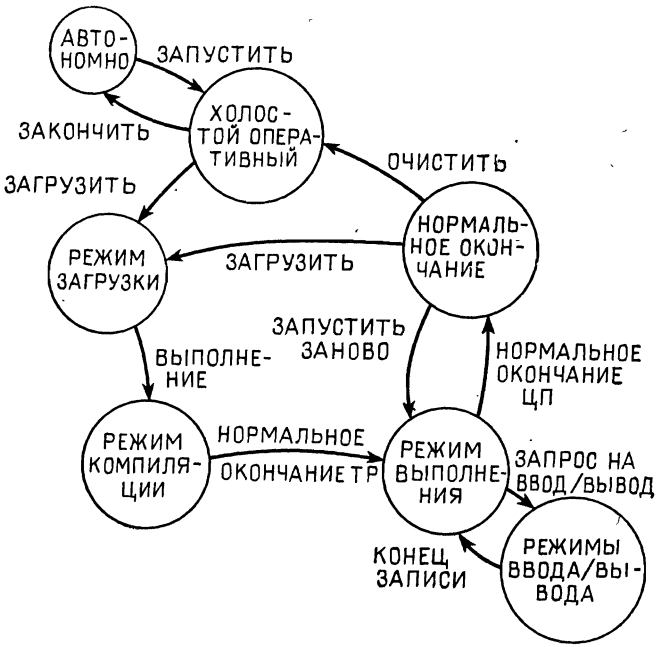


Рис. 14. Идеализированное происхождение задачи с одного терминала.

В следующих разделах разбираются примеры потоков информации во время основных рабочих режимов терминала. Более детальная блок-схема, приведенная на рис. 15, поможет лучше разобраться в описании. Свяжите для себя временную последовательность рабочих состояний терминала на рис. 14 со статической блок-схемой аппаратуры на рис. 15.

Режим загрузки

Режим ЗАГРУЗКА — это режим редактирования текстов ввода/вывода. Его основное назначение — загрузка исходной программы. В обычном случае для записи текста, состоящего из строк, используется отдельный список страниц. Эта область в памяти называется Временной Рабочей Областью (ВРО).

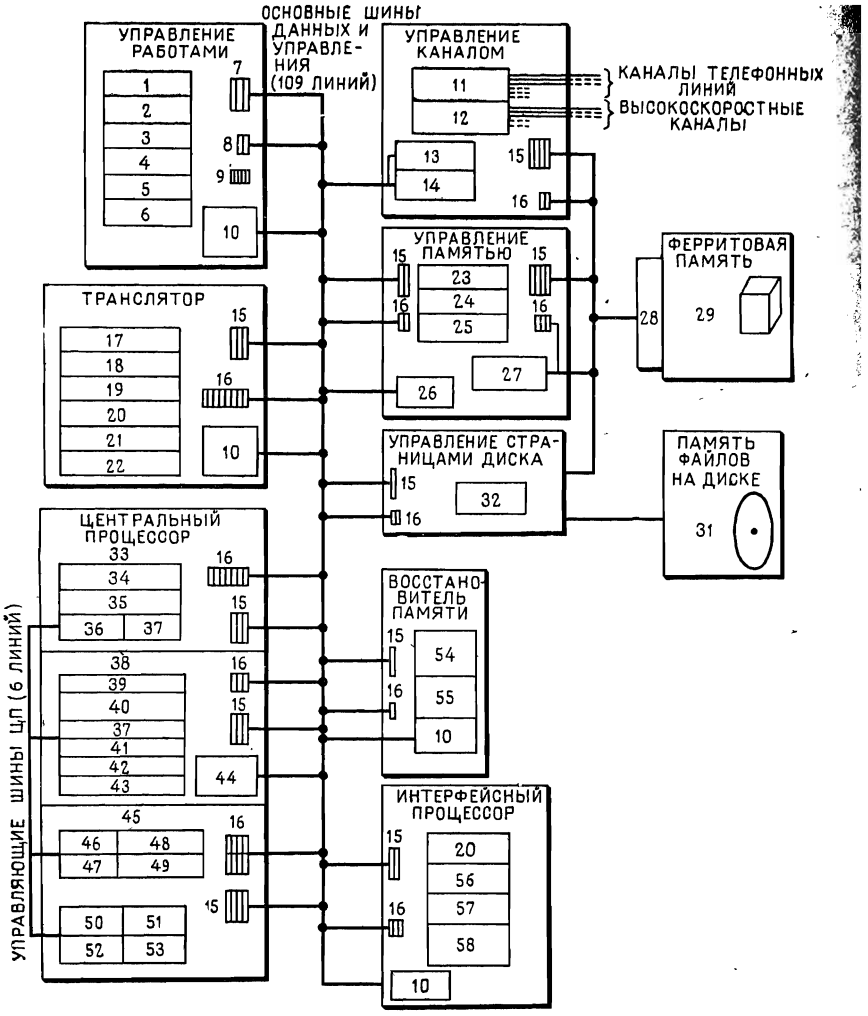


Рис. 15. Более детальная блок-схема системы СИМВОЛ, показывающая конфигурацию регистров и основные функции каждого процессора.

1 — Очереди. 2 — Запуск системы и остановка. 3 — Управление вводом/выводом. 4 — Управление страницами. 5 — Контроль времени. 6 — Контроль ошибок. 7 — Регистры данных. 8 — Адресные регистры. 9 — Регистры связок. 10 — Общее управление. 11 — Приемопередатчики телефонных линий. 12 — Высокоскоростные приемопередатчики. 13 — Управление общим режимом. 14 — Управление высокоскоростным режимом. 15 — Данные. 16 — Адреса. 17 — Синтаксический анализ. 18 — Управление стеком. 19 — Управление компиляцией. 20 — Сканирование символов. 21 — Таблица имен и поиск в ней. 22 — Таблица зарезервированных слов и поиск в ней. 23 — Выдача страниц и их связывание. 24 — Создание групп и их связывание. 25 — Управление таблицей страниц. 26 — Контроль цикла. 27 — Таблица ассоциативных страниц. 28 — Специальная логика интерфейса. 29 — 8192 слова, 64 разряда в слове. Цикл 2,5 мксек. 30 — Шины данных и адресов памяти. 31 — Диск 800 страниц, 120 страниц в сек. Фиксированные головки. 32 — Управление передачами. 33 — Ссылочный Процессор. 34 — Выдача простого адреса. 35 — Выдача адреса по индексу. 36 — Управление оптимизацией. 37 — Присваивание. 38 — Процессор Команд. 39 — Выдача операнда. 40 — Выполнение выражения. 41 — Ввод/вывод. 42 — Управление блоками. 43 — Управление последовательностью выполнения. 44 — Общее управление и управление прерываниями. 45 — Арифметический и Форматный Процессоры. 46 — Сравнение. 47 — Упаковка/распаковка. 48 — Маска. 49 — Формат. 50 — Управление точностью. 51 — Сложение/вычитание. 52 — Сумматор. 53 — Умножение/деление. 54 — Восстановление групп. 55 — Восстановление страниц. 56 — Управление выравниванием. 57 — Управление редактированием. 58 — Генерирование управляющих символов.

Три процессора работают совместно над редактированием текста. Управление Канала (УК) передает символы данных от устройств ввода/вывода в буферы ввода/вывода, находящиеся в главной памяти, и обратно. Когда УК обнаруживает управляющие символы в потоке ввода/вывода, он при помощи цикла обмена передает управляющую информацию в СС. УК является процессором, ориентированным на обработку символьной информации, и может обслужить в режиме переключения до 32 процессоров. УК также имеет высокоскоростной (поблоковый) ре-

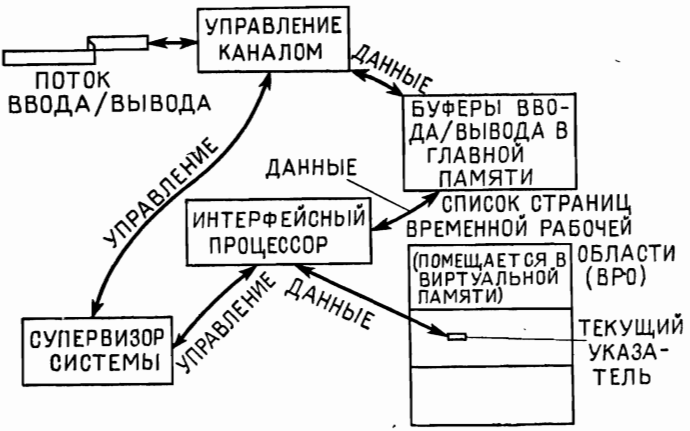


Рис. 16. Потоки информации в режиме ЗАГРУЗКА.

жим работы с приоритетами для обслуживания дисков и высокоскоростных устройств магнитных лент. Поблоковый режим не используется в режиме загрузки и в нормальном режиме ввода/вывода.

Интерфейсный процессор (ИП), работающий во взрывной манере, заполняет или разгружает буферы ввода/вывода и передает соответствующие символы в виртуальную память или в противоположном направлении. При выполнении своих функций ИП работает с текущим указателем текста. ИП выполняет функции для вставки, поиска и показа заданных частей текста, удаления заданных частей текста и перемещения текущего указателя. Основной поток информации во время режима загрузки подытожен на рис. 16.

Оправданием для перенесения функций редактирования в аппаратуру послужило желание освободить ЦП от многих задач, перегружающих систему. Кроме того, если бы УК пришлось связываться непосредственно с виртуальной памятью, то время ответа было бы неприемлемым. ИП был разработан для того, чтобы он осуществлял передачи между небольшими буферами и

страничной памятью. После того как для этой цели был создан специальный процессор, было обнаружено, что многие задачи редактирования и двойной буферизации могут быть решены путем использования той же самой аппаратуры передачи данных.

Процесс, включающий взаимодействие ИП, УК и СС, может быть использован как при подготовке данных в режиме загрузки, так и во время выполнения программы ввода/вывода. Полный набор возможностей редактирования текста доступен для любого оператора программы ввода.

Режим компиляции

Транслятор (ТР) осуществляет компиляцию программы и редактирование связи адресов. ТР получает исходную строку, написанную на входном языке, из ВРО или любой другой области виртуальной памяти, где находится исходный текст. Язык

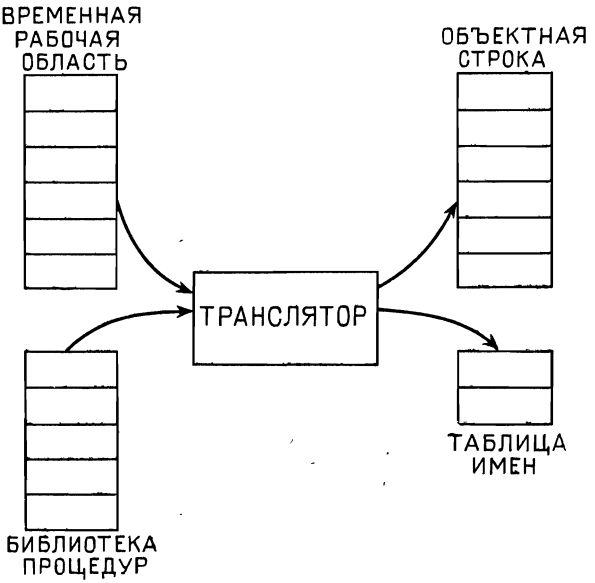


Рис. 17. Потоки информации в режиме КОМПИЛЯЦИЯ.

высокого уровня преобразуется в обратную польскую запись и структурную таблицу имен (идентификаторов). Польская запись, называемая объектной строкой, и таблица имен могут быть записаны в разные или в общий список страниц в виртуальной памяти. Поток информации в режиме трансляции в самом общем виде показан на рис. 17.

ТР осуществляет однопроходную компиляцию, вырабатывая объектную строку, по мере сканирования исходной строки. Во

время сканирования программы по принципу блок за блоком он строит также и таблицу имен. В конце прохода исходного текста ТР вырабатывает таблицу имен и разрешает все глобальные ссылки путем создания соответствующих косвенных связей. Во время прохода по таблице имен создаются ссылки на внешние процедуры. Эти процедуры компилируются и включаются в соответствующие места объектной строки.

ТР находит внешние процедуры в библиотеке в их исходном виде и после компиляции включает их в объектную строку. В библиотеки процедур включены процедуры двух типов — привилегированные и непривилегированные. Привилегированные программы отличаются от обычных тем, что они могут содержать привилегированные операторы прямой работы с памятью, использующие операторы УП. Защита памяти осуществляется за счет проверки привилегированного статуса программ пользователя и программ, к которым они могут обращаться. Непривилегированные программы имеют высокую степень защиты по памяти как от других программ, так и от самих себя, благодаря аппаратному управлению памятью и алгоритмам центрального процессора. Программы, использующие привилегированные операторы, в какой-то мере теряют свою защищенность. Общей защиты памяти вполне достаточно для многотерминальной работы, так как существует контроль за выборкой привилегированных программ и методом их использования.

Режим выполнения

В этом режиме исполнительным устройством является Центральный Процессор (ЦП). Входными данными для него служат оттранслированная с исходного языка строка и таблица имен с учетом вложенных блоков. Так как ЦП работает с языком высокого уровня, а именно с объектной строкой обратной польской записи, то он использует для операндов стек магазинного типа. То есть при выборке данных осуществляется проход по всем косвенным ссылкам до точки, в которой находится адрес памяти, и затем прямая ссылка помещается в стек. Этот процесс продолжается до тех пор, пока в объектной строке не встретится оператор. Каждый оператор берет одну или две (одноместный или двуместный оператор) записи из верхушки стека, производит обработку и результат(ы) помещает в стек.

Обращение к подструктуре, или, что то же самое, выборка переменной по индексу в СИМВОЛе, является гораздо более трудоемкой задачей, чем в обычных системах. Это происходит из-за исключительной динамичности и гибкости этих структур. В обычных системах выборка элемента вектора сводится к простому нахождению базы и индекс-регистра, содержимое которого определяется индексом переменной, и во время выполнения

программы просто делается адресная арифметика для того, чтобы найти желаемый элемент. В СИМВОЛе нет возможности, взяв базовый адрес, произвести вычисление адреса из-за динамической природы распределения места в памяти, а также и потому, что логически последовательные данные не обязательно расположены в памяти последовательно. Отыскание точек, заданных индексами, Ссылочный Процессор (СП) осуществляет в основном методом сканирования, используя некоторые средства для ускорения поиска.

ЦП имеет и еще одно новшество. Все операции обработки производятся над данными переменной длины. Длина строки,

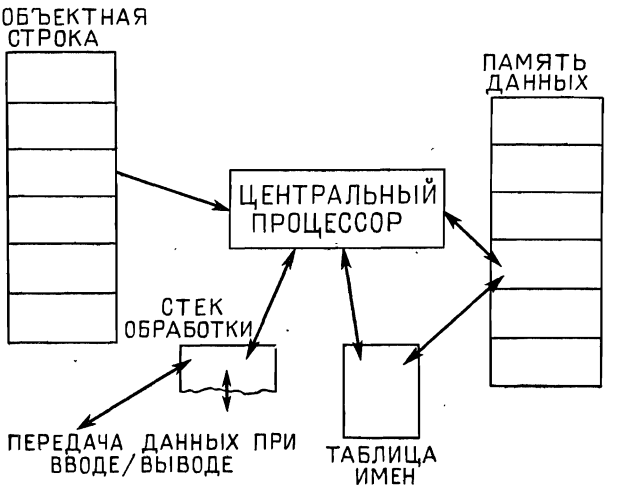


Рис. 18. Потоки информации при выполнении программы.

над которой может быть произведена операция, ограничивается только размером главной памяти. Обрабатываемые числа могут иметь мантиссу длиной до 99 знаков (внутреннее представление — это нормализованные десятичные числа с плавающей запятой). Более того, при помощи регистра ограничения можно задать количество разрядов числа, которое будет обрабатываться. Существует и указатель точности, влияющий на метод вычисления. Обработка чисел, помеченных знаком EM (EMpirical — эмпирическое), ограничивается тем количеством знаков мантиссы, которое содержится в числе, если только на регистре ограничения не установлена еще меньшая величина.

Поток информации для Центрального Процессора показан на рис. 18. ЦП подразделяется на четыре отдельные части: Процессор Команд (ПК), Ссылочный Процессор (СП), Арифметический Процессор (АП), Форматный Процессор (ФП). Как по-

казано на рис. 15, у ЦП есть общие управляющие шины, которые используются для управления различными процессорами во время исполнения программы. В следующих четырех разделах дается функциональное описание каждого из процессоров, входящего в ЦП.

Процессор Команд

Процессор Команд (ПК) — это основная управляющая и переключающая часть ЦП. В задачу ПК входит сканирование объектной строки и накапливание в стеке единиц информации для различных устройств, которые он обслуживает. Например, если во время накопления операндов для обрабатывающего устройства обнаружится, что требуется какое-либо преобразование типа данных, то ПК запрашивает ФП для проведения преобразования. Подобным же образом ссылка на структуру и все ее индексы вычисляются и помещаются в стек, который затем передается СП для обработки.

ПК также подготавливает данные для присваивания, которое производит СП, и данные для вывода на устройства ввода/вывода. В первом случае ПК помещает в стек ссылку присваивания и данные, а во втором случае помещает данные в стек и возвращает управление системе.

Динамическое создание вложенных языковых блоков — еще одна важная функция ПК. Если это понятие незнакомо читателю, то отсылаем его к статье о языке СИМВОЛ [3]. Коротко, блоки — это языковые конструкции, состоящие из программных сегментов, заключенных между зарезервированными словами BLOCK и END (PROCEDURE и ON также создают блоки). Внутри блока все использования идентификатора локальны по отношению к этому блоку, если только нет оператора GLOBAL, и поэтому для каждого блока создается своя таблица имен. Общая структура таблицы имен имеет как статическую сторону, определяемую тем, как написана программа, так и динамическую сторону, определяемую тем, в каком порядке блоки выполнялись. Это последняя из особенностей программы, которой мы касаемся в данном описании. Как только ПК встречает новый блок, обработка старого блока приостанавливается и вся информация об этом блоке, которая должна быть сохранена для последующей активизации блока (иногда ее называют записью активизации), отправляется в стек. После этого формируются новый стек и его запись активизации на верхушке старого стека. Конечно, новая запись должна содержать ссылку на старую запись, для того чтобы после окончания нового блока можно было восстановить старый блок вместе с информацией о его состоянии.

Дополнительные трудности возникают, если встречаются процедурные блоки, из-за того, что приходится устанавливать

соответствие между фактическими и формальными параметрами (снова см. статью [3]). ПК передает из объектной строки в стек ссылки на фактические параметры и выбирает таблицу имен нового блока, в первых строках которой находятся формальные параметры. Затем ссылки на фактические параметры одна за другой помещаются в строки формальных параметров таблицы имен. На этом связывание параметров заканчивается, и остальные действия для процедуры выполняются так же, как и для обычного блока. Когда ПК встречает в таблице имен строку, помеченную как формальный параметр, он при помощи косвенной адресации вызывает на ее место фактический параметр, который может быть переменной, константой, меткой, литералом, процедурой или выражением, но не может быть оператором. Механизм косвенной адресации используется и при работе с этим стеком ПК. В стек помещается очень небольшое количество информации о состоянии. В основном это адрес объектной строки, указывающий, в каком месте было приостановлено ее выполнение. После этого исполняется новая объектная строка фактического параметра, которая пользуется стеком до тех пор, пока не встретится оператор возврата, указывающий на конец строки фактического параметра. Тогда восстанавливается предыдущее состояние, информация о котором хранилась в стеке, и продолжается исполнение прерванной объектной строки, при котором используются результаты, получившиеся после выполнения строки формального параметра и находящиеся на вершине стека.

Ссылочный Процессор

Основной задачей СП является работа со структурами. Дополнительно на него возложена работа по выборке из таблиц имен ПК адресов единиц информации. А именно, встретив в объектной строке адрес, ПК передает его СП вместе с запросом «найти простой адрес». СП производит некоторые действия, зависящие от типа идентификатора. Если единица данных уже существует, то СП выдает адрес данного вместе с кодом, определяющим его тип. Если этого данного еще нет, то СП, перед тем как выдать адрес, сначала отводит для этого данного место в памяти. Подобным же образом СП выдает ссылки на метки и процедуры, а если какие-нибудь идентификаторы оказываются глобальными, то он, прежде чем вернуть ссылку, прослеживает на требуемую глубину глобальные косвенные ссылки. Любые аномалии в таблице имен приводят к тому, что выдается сообщение об ошибке.

Задача работы со структурами может быть разбита на две части: создание структур и подструктур и создание ссылок на точки подструктуры. Напомним, что структуры динамически изменяются по всем своим характеристикам. Поэтому при созда-

нии структур имеется два дополнительных этапа: создание базовых структур и перестройка подструктур. Дополнительно для ссылки на подструктуру в языке имеется возможность индексации символов. В этом случае последним индексом может быть «граничная пара» индексов, которая указывает начальную точку и количество символов подполя в поле, заданном предыдущими индексами.

СП строит структуру из ее линейного представления, находящегося в стеке ПК. СП должен записать эту структуру в память, заменяя ее линейную форму на иерархическую со ссылками в верхнем уровне на элементы более низкого уровня (см. рис. 10, 11 и 13). Это преобразование происходит следующим образом. СП отводит новую группу памяти каждый раз, когда встречается новый левый знак группы, и, отведя строку для новой группы в группе более высокого уровня, заполняет группу элементами, сохраняя ссылку на группу более высокого уровня в своем собственном стеке групповых ссылок. Всякий раз, когда в стеке ПК встречается правый знак группы, текущая группа памяти закрывается пометкой «конец вектора» и из стека групповых ссылок выбирается следующая позиция группы памяти более высокого уровня. Этот процесс продолжается до тех пор, пока не будет исчерпана вся структура, находящаяся в стеке ПК, и в результате не будет получена иерархическая структура со ссылками.

Таким же образом осуществляется привязывание новой структуры к существующей точке подструктуры. Старая структура уничтожается (занимаемое ею место в памяти будет позднее обработано Восстановителем Памяти), и новая линейная структура, находящаяся в стеке, структуризуется и привязывается к соответствующей точке подструктуры. Разрешаются все комбинации замены: структура на структуру, поле на поле, структура на поле и поле на структуру. Во втором случае, когда поле заменяется на поле, могут возникнуть сложности в том случае, если новое поле больше, чем старое, так как это ведет к расширению вектора (в противном случае свободное место заполняется нулями). Простое решение, заключающееся в создании неиерархической ссылки на новое место в памяти, не дает хорошего результата, если одно за другим расширяются последовательные слова большого вектора. Решение состоит в том, чтобы использовать новую группу памяти только после того, как будет установлено, что в текущей и последующей группах не осталось необходимого свободного места, а после выдачи новой группы памяти излишек, получившийся в результате расширения, переписать целиком на новое поле. В этом случае при расширении многих полей вектора используется заново отведенное место в памяти.

Общий алгоритм хождения по структуре заключается в том, что СП просматривает стек ПК в обратном порядке, пока не найдет ссылки на структуру, и затем движется вперед последовательно по каждому индексу, выбирая каждый вектор и используя соответствующие методы ускорения, пока не дойдет до последнего индекса. В этой точке СП заменяет индексированную ссылку в стеке УП на простую ссылку, которая указывает на подструктуру или на поле, если был достигнут уровень данных. При обработке индексированной ссылки по предварительно построенной структуре в любой точке может оказаться, что структура не распространяется до точки, на которую указывает ссылка (ошибка по диапазону индексации). Правила языка в этом случае предусматривают, что должно быть отведено новое место, которое требуется, чтобы реализовать структуру до точки, на которую указывает значение индекса (промежуточные свободные поля заполняются нулями). Эта работа выполняется СП.

Если после задания ссылки на структуру до уровня поля в стеке ПК встречается граничная пара индексов, то СП, сканируя поле, выбирает нужные символы в заданном количестве и помещает результат в стек ПК. В этом случае, если граничная пара встретилась раньше, чем был достигнут уровень поля, фиксируется ошибка.

Арифметический Процессор

АП — это устройство последовательной обработки данных переменной длины, представляющих собой десятичные нормализованные числа с плавающей запятой. Последовательность выполнения операций — от старших разрядов к младшим. Такой порядок выполнения операций упрощает обработку данных, так как позволяет использовать сходные операции над регистрами как для строк, так и для чисел. Кроме того, в этом случае операция сравнения выполняется быстрее, так как результат определяется по первому несовпадению цифр. Аппаратура обработки обладает еще двумя важными свойствами: есть регистр ограничения, загружаемый ПК по командам языка и завершающий обработку при достижении заданной точности, и есть указатель точности, который может быть в каждом операнде и определять его действительную точность и таким образом управлять точностью результата.

Процессором выполняются операции сложения, вычитания, умножения и деления. При сложении и вычитании один или два операнда проходят через устройство (от старших разрядов к младшим) до тех пор, пока их порядки не выравниваются, и с этого момента оба операнда начинают следовать через устрой-

ство синхронно. Так как число есть величина со знаком, то по комбинации знаков операндов и знака операции нужно принять решение, какой из операндов, если в этом есть необходимость, подавать в дополнительном коде. Из-за того, что арифметические операции выполняются последовательно, начиная со старших разрядов, требуется счетчик девяток [10], который задерживает выход строки девяток до тех пор, пока не станет известно, будет перенос или нет. В конце концов либо встретится эмпирический конец операнда, либо будет достигнута величина, заданная на регистре ограничения, либо оба точные числа исчерпаются. В этом месте работа АП заканчивается и управление возвращается ПК.

Умножение выполняется как ряд последовательных сложений или вычитаний со сдвигом, до тех пор пока не окончатся разряды множителя. Только после того, как будет получена полная трапеция всех частных произведений, производится округление для достижения заданной точности. Если цифра множителя, на которую производится умножение, больше чем четыре, то производится ускоренное прибавление единицы к предыдущей цифре множителя и вычитание из частичного произведения. Конечно же, при умножении (и делении) порядки складываются (вычитаются) и никакого сдвига мантиссы не требуется. Деление выполняется как ряд последовательных вычитаний без восстановления остатка до тех пор, пока точность результата не станет равна точности менее точного из двух операндов или содержимому регистра ограничений.

Так как обработка в этой системе проводится последовательно в десятичной форме с не слишком эффективными методами ускорения, то скорость обработки сильно зависит от размера операндов. Когда в счетчик ограничений заслана небольшая величина, скажем 5, обработка может быть очень быстрой, но деление 99-разрядных чисел идет крайне медленно. Поэтому важно, чтобы пользователь выбирал точность, не превышающую той, которая ему нужна.

Сравнение чисел выполняется в АП как операция вычитания, но оканчивается она, как только будет обнаружено несовпадение, и в качестве результата будет отправлен нуль. В задачу ПК входят комбинирование результата, возвращенного из АП, с заданной операцией сравнения и выработка конечного результата для стека ПК.

Форматный Процессор

Устройство ФП осуществляет операцию СЦЕПЛЕНИЯ (JOIN) строк, двоичные операции над строками И (AND), ИЛИ (OR), НЕ (NOT), операции сравнения строк ДО (BEFORE), ТО

ЖЕ (SAME), ПОСЛЕ (AFTER), операции ФОРМАТ (FORMAT) и МАСКА (MASK), а также автоматическое преобразование типов операндов, необходимое для ПК: число в строку, строку в число и число в целое (используемое в основном при индексации). Все эти операции так же выполняются последовательно. Операция СЦЕПЛЕНИЯ очевидна. Она приписывает второй операнд к концу первого, образуя в результате единый операнд.

Двоичные операции выполняются посимвольно, с применением заданной операций, в результате чего получаются символы 0 или 1. Более короткий операнд дополняется нулями.

Операция сравнения строк также выполняется посимвольно. Символы сравниваются последовательно до тех пор, пока не будет обнаружено несоответствие (порядок символов определяется ASCII, и в результате возвращается либо 0, либо 1.)

Операции ФОРМАТ и МАСКА представляют собой мощное средство работы со строками для широкого круга применений, начиная от подготовки платежных ведомостей и банковских форм и кончая операциями в системе программного обеспечения для символьных преобразований. Операция ФОРМАТ преобразует упакованное число в строку, так что позволяет пользователю описать формат результата в виде наглядной строки символов. Операция выполняется последовательно в соответствии с видом операндов. Стандартное преобразование из упакованной числовой формы в строку по умолчанию является частью операции ФОРМАТ. Операция МАСКА аналогична операции ФОРМАТ с той разницей, что происходит преобразование из строки в строку. МАСКА может быть использована для вставки и выбрасывания символов и для управления разрядкой (пробелы). Она часто используется для управления длиной поля и ее измерения. И при этой операции символы обрабатываются последовательно.

СУПЕРВИЗОР СИСТЕМЫ

Нагрузка, компиляция, исполнение и ввод/вывод представляют собой основные режимы обработки в системе. Для терминалов существуют три дополнительных режима работы: автономный, холостой оперативный и режим нормального окончания. Все эти режимы пассивные и различаются только тем, какие операторы изменения состояния могут быть выданы в них по прерыванию. Например, состояние нормального окончания является единственным состоянием, в котором может быть выполнена команда НАЧАТЬ ЗАНОВО (RESTART). НАЧАТЬ ЗАНОВО может быть выполнена только в том случае, если объектная строка осталась в состоянии, допускающем ее повторное использование.

На рис. 14 показано несколько переходных состояний терминала. Важно, что все переходные состояния поддерживаются аппаратными алгоритмами. Когда СС получает управляющий код, соответствующий оператору ВЫПОЛНИТЬ (RUN), то переход из режима загрузки в режим компиляции может произойти без вмешательства программного обеспечения. Есть много других переходных состояний, но они, как правило, требуют взаимодействия с программным обеспечением системы. Переход из режима

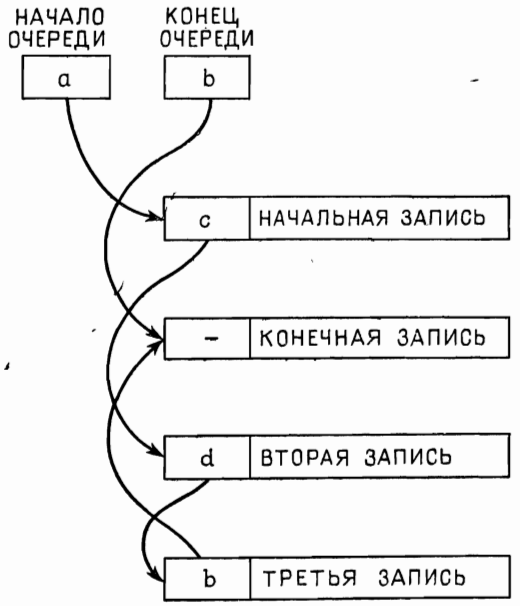


Рис. 19. Типичная структура очереди задач.

загрузки в режим компиляции включает в себя следующие шаги. В случае если ИП находится в активном состоянии, ему должна быть предоставлена возможность закончить работу, чтобы исходная строка ввелась полностью. Задача затем удаляется из очереди в ИП и передается в очередь к ТР. Кроме того, в управляющие таблицы, находящиеся в главной памяти, засылаются начальные значения, что делает доступным для ТР адрес начала исходной строки и адреса библиотек процедур, которые будут использоваться.

Типичная очередь задач показана на рис. 19. Она состоит из связанного списка записей (управляющих слов). Очередь имеет два указателя — один на начало и другой на конец списка. При помощи этих указателей легко добавлять позиции как к началу, так и концу очереди.

Каждый раз при возникновении управления переходным процессом СС производит соответствующие действия по добавлению и выбрасыванию, обновляя очереди к каждому устройству, участвующему в работе. Это часть первой фазы работы СС по обработке любой задачи. Вторая фаза работы СС включает в себя распределение работ по свободным процессорам, которые имеют в своих очередях соответствующие задачи.

Алгоритм многопроцессорной обработки в основном включает в себя работу с очередями к ЦП, ТР, ИП, ВЦ и УД и использование этих очередей. В СС имеется универсальный процессор для работы с очередями, который позволяет добавлять в начало или конец или выбрасывать записи из любой очереди.

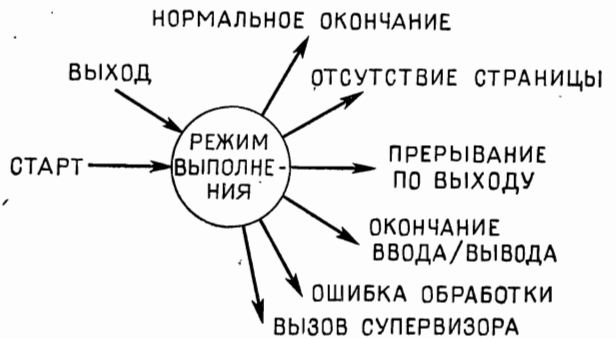


Рис. 20. Переходные состояния Центрального Процессора.

Алгоритм имеет режим отказа, который целиком реализуется аппаратурой. Различные параметры, влияющие на динамику работы, могут быть установлены программным обеспечением. Например, для каждой записи в очереди к ЦП имеется два счетчика времени. Один из них измеряет общее время обработки, а второй измеряет действительное время нахождения задачи на вершине очереди. Величины, устанавливаемые на эти счетчики, задаются как параметры при установке задачи в очередь. Когда эти величины уменьшаются до нуля, генерируется задача СС, которая обеспечивает изменение очередей. В большинстве случаев это приводит к тому, что задача из позиции с высоким приоритетом вблизи начала очереди перемещается в позицию с низким приоритетом вблизи конца очереди.

Для пояснения общих принципов распределения работ в системе на рис. 14 в сильно упрощенном виде показана последовательность обработки в системе. На рис. 20 показаны управляющие команды, получаемые и выдаваемые Центральным Процессором. СС может приказать ЦП начать задачу или прекратить работу с ней. Как видно, ЦП может прекратить обработку

данной задачи по одной из шести основных причин. Рассмотрим окончание ввода/вывода. В основном при управлении вводом/выводом для большинства терминалов достаточно аппаратного алгоритма. С другой стороны, если желательно, чтобы терминал работал в режиме пакетной обработки с устройством ввода/вывода боббинного типа, то в этом случае необходимо, чтобы управление вводом/выводом было передано процедуре программного обеспечения системы. Для вывода программного обеспечения терминал, требующий обслуживания ввода/вывода, должен установить специальный управляющий разряд в управляющем

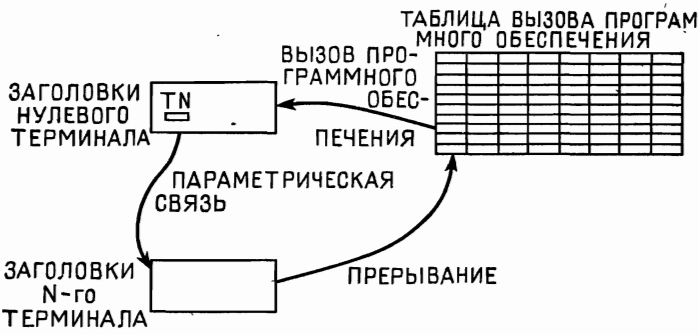


Рис. 21. Механизм вызова программного обеспечения по прерыванию изменения состояния.

слове терминала для этого канала. Тогда в СС автоматически возникает вызов программного обеспечения.

Вызов программного обеспечения в СИМВОЛе осуществляется запуском псевдотерминала, использующего в качестве параметра номер требуемого канала. Таким образом управляющие таблицы заголовка запрашиваемого канала могут быть обработаны как данные. Это показано на рис. 21, где прерывание специального класса приводит к выбору нужной программы, которая определяется по таблице вызова программного обеспечения. Управление передается псевдотерминалу с параметром TN. Различные классы прерываний размещены в различных управляющих словах таблицы управления программного обеспечения. Поэтому только необходимые процедуры программного обеспечения будут выбираться в виртуальную память. В СИМВОЛе через таблицу управления программным обеспечением, расположенную в конце главной памяти, может быть вызвано свыше 80 программных прерываний. В этом и заключается основа интерфейса между аппаратурой и программным обеспечением.

ЗАКЛЮЧЕНИЕ

Традиционная граница между аппаратурой и программным обеспечением за последние десять лет была во многом нарушена, и в дальнейшем этот процесс будет идти еще интенсивней. Мы верим, что СИМВОЛ является важным шагом на пути к созданию аппаратуры, обладающей большими возможностями.

Система СИМВОЛ сейчас находится в той фазе, когда ее каждодневное использование позволяет оценить систему всесторонне и выявить ее сильные и слабые стороны. Разработчики уже смогли оценить достоинства выбранных принципов. Общий подход к организации и управлению памятью можно рассматривать как очень важный шаг вперед. Перемещение атрибутов данных из команд к данным можно считать принципиальным.

Мы не утверждаем, что система СИМВОЛ оптимально сбалансирована по отношению к производительности и использованию аппаратуры. В некоторых критических местах управления и распределения памяти, а также при распределении ресурсов системы полученная эффективность в 10—100 раз выше обычной. Некоторые аспекты работы со структурами являются большим достижением по сравнению с процессорами, работающими с программными списками, но такого преимущества нет, когда идет обращение к некоторым типам больших массивов. Многие недостатки первой модели СИМВОЛа были обнаружены слишком поздно, чтобы усовершенствования могли быть включены в существующую аппаратуру. Многие другие качества системы, такие, как страничная организация памяти и алгоритмы распределения ресурсов в системе, могут быть оценены только после накопления достаточного опыта в процессе эксплуатации системы.

Специалисты в области вычислительной техники в течение многих лет спорили по вопросам: Можно ли сделать аппаратный компилятор? Может ли сердце системы, ее супервизор, быть зафиксирован в аппаратуре? Может ли аппаратура заниматься управлением и распределением памяти данных? Можно ли разработать аппаратуру, которая взяла бы на себя основные функции программного обеспечения? Можно ли отладить сложную аппаратуру? С запуском системы СИМВОЛ на эти и многие другие вопросы получен положительный ответ. Самое большое значение всего проекта состоит в том, что общие понятия представлены в виде полномасштабной работающей машины.

Кроме авторов этой статьи, многие другие сотрудники принимали участие в разработке некоторых вопросов системы СИМВОЛ. Решающими факторами в завершении системы были дух сотрудничества и преданность, продемонстрированные на протяжении всей разработки проекта всеми членами коллектива.

Авторы хотят поблагодарить Джорджа Пауэрса, Стенли Мейзера и Рассела Бриггса за их участие в разработке аппаратуры, а также Гамильтона Ричардса и госпожу Хилму Мортелл за их участие в начальной стадии разработки программного обеспечения.

ЛИТЕРАТУРА

1. Rice R., Smith W. R., SYMBOL: A major departure from classic software dominated von Neumann computing systems, *Proc. AFIPS*, **38** (1971) SJCC.
2. Mazor S., Programming and/or logic design, Digest of the 1968 Computer Group Conference June 1968 Los Angeles.
3. Chesley G. D., Smith W. R., The hardware-implemented high-lvel machine language for SYMBOL, *Proc. AFIPS*, **38** (1971) SJCC.
4. Cowart B. E., Rice R., Lundstrom S. F., The physical attributes and testing aspects of the SYMBOL system, *Proc. AFIPS*, **38** (1971) SJCC.
5. Smith W. R., System design based on LSI constraints: A case history, Digest of the 1968 Computer Group Conference June 1968 Los Angeles.
6. Kilburn T., One-level storage system, *IRE Transactions on Electronic Computer*, EC-11, № 2 (April 1962).
7. Glaser E. L., Couleur J. F., Oliver G. A., System design of a computer for time sharing applications, *Proc. AFIPS*, **27**, part 1, 1965 FJCC.
8. Corbato F. J., Vyssotsky V. A., Introduction and overview of the maltics system, *Proc. AFIPS*, **27**, part 1, 1965 FJCC.
9. Smith W. R., Associative memory techniques for large data processors, PhD Dissertation Iowa State University, 1963.
10. Mullery A. P., Schauer R. F., Rice R., ADAM: A problem-oriented symbol processor, *Proc. AFIPS*, **23** (1963) SJCC.

Автоматический анализ текстов и поиск документов[1])

Джерард Сэлтон

1. ЛИНГВИСТИКА И ОБРАБОТКА ИНФОРМАЦИИ

В результате многих лет работы специалисты в области вычислительной лингвистики и автоматической обработки текстов пришли к убеждению, что наиболее трудные проблемы, возникающие в этих областях, по-видимому, не могут быть решены без привлечения мощных средств лингвистического анализа. Таким образом, широко распространено убеждение, что первоочередной задачей должно стать изучение структурных и семантических свойств естественного языка в надежде достичь такого их понимания, которое привело бы к решению наиболее трудных проблем автоматической обработки текстов [1, 2]. К сожалению, как отмечают Пекак и Пратт, вероятность достичь в обозримом будущем необходимой степени владения структурными свойствами естественного языка очень мала, так как «до сих пор явно видны очень существенные недостатки в автоматическом анализе текстов; все существующие системы и/или модели малы и ориентированы на решение некоторой ограниченной задачи...; нет систем, рассматривающих весь язык, а не некоторое его подмножество...; очень мало таких систем, в которых делается попытка выйти при анализе за границы предложения...; все упомянутые теоретические подходы в значительной степени носят предварительный характер» [3].

Таким образом, складывается довольно тяжелое положение: с одной стороны, говорится, что для решения задач, возникающих при обработке текстов на естественном языке, необходимо лучшее понимание синтаксиса и семантики языка; с другой стороны, перспектива достижения такого понимания, которое должно обеспечить значительные и быстрые успехи вычислительной лингвистики, является отдаленной.

Однако при детальном рассмотрении различных задач автоматической обработки текстов становится ясно, что несколько преждевременно предсказывать полный провал при решении этой задачи. Во-первых, разнообразны применения методов автоматической обработки текстов, которые отличаются по сложности и объему задач. Во-вторых, нет никаких оснований пола-

[1]) Salton G., Recent studies in automatic text analysis and document retrieval, *J. of the Ass. for Comp. Mach.* **20**, № 2 (1973).

гать, что недостающие средства лингвистического анализа не могут быть чем-либо заменены.

Тогда, возможно, стоит подойти к решению некоторых задач автоматической обработки текста, возникающих при информационном поиске, пытаясь использовать либо экстралингвистические процедуры, не опирающиеся на структуру и семантику предложения, либо простые лингвистические средства, с помощью которых можно устранить наиболее неприятные осложнения. Эти два пути исследуются ниже.

2. УПРОЩЕННЫЙ АНАЛИЗ ТЕКСТА

Первый подход к проблеме анализа текста состоит в том, чтобы оставить в стороне многие из трудностей, относящиеся к области семантики, например многозначность слов, распознавание синонимов, рассмотрение знаков пунктуации, интерпретацию анафорических, эллиптических и идиоматических конструкций, проблему косвенных ссылок, анализ связного текста в пределах абзаца или большего отрезка текста и т. п. Вместо этого особое внимание уделяется структурным свойствам языка.

К сожалению, существующие программы синтаксического анализа не так уж удобны для использования, поскольку не для каждого предложения получается единственный правильный результат анализа. Однако существующие анализаторы, которые представляют предложение в виде так называемых «непосредственных составляющих», с достаточной точностью находят в тексте определенные словосочетания, такие, как именные, предложные группы, группы вида «подлежащее — сказуемое — дополнение». Поэтому некоторые исследователи делали попытки включить программы выделения простых словосочетаний в программы, предназначенные для автоматического индексирования или реферирования текстов [4—7]. Эти программы можно использовать многими способами. Чаще всего процесс автоматического индексирования складывается из следующих шагов [4]:

1) Сначала из текста выделчются отдельные слова.

2) Отождествляются некоторые синтаксические категории, такие, как определения, предлоги, подчинительные союзы.

3) Образуются словосочетания, представляющие собой цепочку слов, начинающуюся с одной из уже определенных синтаксических категорий, например с предлога или относительного местоимения, и заканчивающуюся перед началом следующей уже определенной синтаксической категории.

4) Устанавливаются связи между элементами различных словосочетаний; главным образом находятся антецеденты определенных местоимений и хозяева предложных групп.

5) Словосочетания, обладающие определенными структурными свойствами, рассматриваются как идентификаторы содер-

жания, причем к ним относятся главным образом существительные и прилагательные с некоторыми определенными предлогами.

Было проделано несколько экспериментов по оценке систем автоматического индексирования, которые содержали процедуры, аналогичные вышеописанным. Вообще говоря, оказывается, что значительная часть выражений, выделенных автоматически, была бы выбрана и человеком, перед которым стояла бы та же задача [8]. С другой стороны, если сравнить алгоритм выделения словосочетаний, в котором используется упрощенный синтаксический анализ, с другими, более простыми методами анализа содержания, куда не входит ни синтаксический, ни семантический компонент, получается удивительная вещь, а именно: без использования синтаксического компонента получаются в среднем лучшие результаты, чем при его использовании. Например, Клевердон пишет в [9]: «Крэнфилдский эксперимент показывает, что унитермные языки индексирования превосходят языки любого другого типа...; языки индексирования, в которых используются словосочетания, оказались слишком специфичными для текстов на естественном языке».

Неудачу, которая постигла исследователей, когда они пытались применить простые средства синтаксического анализа при информационном поиске, можно объяснить следующими соображениями. Самое очевидное — это неудачный выбор анализатора, который выделяет словосочетания. Возможно, что не были выделены многие нужные словосочетания, в то время как выделенные словосочетания могли ничего не давать для анализа содержания. Во-вторых, в системах, рассчитанных на разнородный состав потребителей, нецелесообразно слишком подробно описывать содержание документа; в частности, менее точное описание содержания документа с помощью отдельных понятий могло больше соответствовать интересам потребителей, чем более подробное описание с помощью словосочетаний, поскольку довольно трудно учесть заранее интересы всех потребителей. Наконец, большинство систем, оценивающих результаты работы ИПС, используют критерии эффективности, усредненные по многим запросам. Тогда ситуация, при которой достигается эффективность, бóльшая, чем средняя, могла оказаться более предпочтительной, чем ситуация, при которой одни потребители дают системе очень высокие оценки, а другие — очень низкие.

Так или иначе, в результате рассмотрения имеющихся результатов, вероятно, можно было бы сделать вывод о том, что существующие алгоритмы анализа текста, которые можно использовать в системах автоматической обработки текстов, не дают результатов, которые оправдывали бы усилия, затраченные на их разработку и внедрение. Полный анализ текста не-

возможен сейчас, потому что нет еще необходимого для этого понимания структуры языка; простые же методы анализа, опирающиеся на синтаксис, не дают достаточно хороших результатов.

Остается только выяснить, можно ли воспользоваться какими-либо нелингвистическими, машинными средствами, способными в какой-то степени восполнить пробел в наших знаниях о языке, а также пригодными для выполнения интеллектуальной работы, осуществляемой человеком при вводе документов. Этот вопрос исследуется в следующем разделе статьи.

3. СРАВНЕНИЕ РУЧНОГО И АВТОМАТИЧЕСКОГО СПОСОБОВ ИНДЕКСИРОВАНИЯ

За многие годы накопилось большое количество материалов, отражающих результаты сравнения автоматического и ручного методов индексирования [8—18]. Если рассматривать ключевые слова, выделенные автоматически, то оказывается, что в 60—80% случаев они совпадают с терминами, вручную приписываемыми специалистом в рассматриваемой предметной области. Более того, результаты поиска, проведенного по документам, заиндексированным автоматически, в среднем не отличаются значительно от результатов, полученных в случае обычного, ручного индексирования. Как указывает Свонсон [18]: «Если от машин можно ждать, по-видимому, только частичного успеха в индексировании документов, ... то от людей не следует ожидать даже этого».

Именно для того, чтобы выяснить, могут ли полностью автоматические методы обработки текста соперничать в эффективности с обычными поисковыми операциями, осуществляемыми людьми, а также для того, чтобы найти автоматические методы, наиболее перспективные в этом отношении, в течение нескольких последних лет были проведены эксперименты по сравнению информационно-поисковой системы MEDLARS, работающей в Национальной медицинской библиотеке в Вашингтоне, с экспериментальной системой SMART.

В системе MEDLARS осуществляются обычные операции индексирования: всем поступающим документам и запросам специалисты в определенной предметной области приписывают ключевые слова, или термины, используемые при индексировании [19]. Список употребляемых при этом слов контролируется с помощью опубликованного тезауруса, известного под названием Mesh (Medical Subject Headings). Поиск осуществляется путем сравнения списка ключевых слов, приписанных данному документу, с запросом, представленным в виде булевой формулы, состоящей из терминов. Точнее, в ответ на соответствующий за-

прос выдаются все те документы, которые содержат в своем описании подходящую комбинацию ключевых слов, в то время как документы, не содержащие этой комбинации, остаются в памяти машины. Как и в любой ИПС, использующей ключевые слова, процесс поиска просто разделяет все документы на два множества — выданные документы и невыданные документы. Не делается никакого разбиения документов по рангам ни для выданных документов, ни для оставшихся, поэтому потребитель, по всей вероятности, не имеет никакого удобного способа выделения наиболее полезных для него документов среди потенциально очень большого массива выданных документов.

Напротив, система SMART работает без какого бы то ни было анализа содержания документа, осуществляемого вручную [20, 21]. Какие-то части текста документа, отражающие его содержание, — обычно рефераты — вместе с текстами запросов вводятся в машину. Затем работают алгоритмы автоматического анализа текста, в результате работы которых для каждого документа и запроса получается некоторый «вектор понятий», содержащий термины, или понятия, которые отражают содержание документа, вместе с приписанными им весами. Обычно для описания некоторого документа используется около 100 различных понятий. Затем идет сравнение векторов запроса и документа, при этом для каждой пары документ — запрос вычисляется коэффициент близости, или коэффициент корреляции. Потребителю документы выдаются в порядке уменьшения соответствующих коэффициентов близости. Таким образом, потребитель может взглянуть, например, только на первый выданный документ — этот документ считается наиболее релевантным запросу — или на первые 5—10 документов.

Для оценки эффективности автоматических способов обработки текста, применяемых при информационном поиске, особенно уместно сравнить между собой системы SMART и MEDLARS по следующим соображениям. Система MEDLARS представляет собой широко известную ИПС с общепринятыми методами поиска, которая действует уже много лет и оперирует несколькими сотнями тысяч документов. В системе же SMART применяются автоматические средства анализа текста и итеративный поиск, контролируемый потребителем, т. е. делается попытка заставить машину выполнять интеллектуальную работу по вводу документов, которую делают обычно опытные индексаторы и документалисты. Таким образом, эти две системы особенно ярко отражают тот уровень, который достигнут в этих двух направлениях.

В последующих разделах описывается организация экспериментов по сравнению систем SMART и MEDLARS, а также приводятся основные полученные результаты.

4. ЭКСПЕРИМЕНТЫ ПО СРАВНЕНИЮ СИСТЕМ SMART И MEDLARS

4.1. Введение

Если мы хотим, чтобы результаты, полученные в ходе эксперимента, можно было считать достоверными, необходимо поставить обе рассматриваемые системы в одинаковые условия, а также снабдить систему SMART теми же операционными характеристиками, которые присущи системе MEDLARS. Тогда должны выполняться следующие основные условия:

1) Запросы, которые будут использоваться в эксперименте, должны быть реальными запросами потребителей системы MEDLARS, уже обработанными этой системой.

2) Экспериментальный массив документов должен состоять из документов, первоначально входивших в банк данных системы MEDLARS и выбранных таким образом, чтобы в эксперименте не использовались никакие предварительные сведения о том, будет ли выдан любой данный документ какой-либо из систем.

3) Максимальное число документов, которое должно быть выдано в ответ на некоторый запрос системой SMART (значение предела выдачи), должно соответствовать пределу выдачи в системе MEDLARS, с тем чтобы для оценки обеих систем использовалось одно и то же число документов.

Параметрами, с помощью которых в настоящей работе оценивается эффективность поиска, служат широко известные понятия точности и полноты, которые в прошлом широко использовались в ряде подобных исследований. Полнота определяется как доля выданных релевантных документов среди всех релевантных документов, а точность — как доля релевантных документов среди всех выданных. Точнее,

$$\text{полнота} = \frac{\text{число выданных релевантных документов}}{\text{число релевантных документов во всем массиве}}, \quad (1)$$

$$\text{точность} = \frac{\text{число выданных релевантных документов}}{\text{число всех выданных документов}}. \quad (2)$$

В идеале хотелось бы выдавать все релевантные документы, не получая при этом ни одного лишнего. В таком случае полнота и точность достигают своего предельного значения и равны 1. На практике, как правило, получаются результаты, далекие от идеальных, при этом значения полноты и точности значительно меньше единицы. Кроме того, обычно не приводятся оценки эффективности системы для каждого отдельного запроса. Вместо этого, для того чтобы отразить уровень эффективности при работе системы с произвольным заданным запросом, по

большому числу запросов вычисляются некоторые средние значения.

Чтобы вычислить значения полноты и точности, нужно для каждого данного запроса располагать сведениями не только о выданных документах, но и о всех документах в поисковом массиве, релевантных данному запросу. Для небольших экспериментальных массивов вроде тех, которые используются в данном исследовании, довольно часто можно получить исчерпывающие сведения о том, какие документы массива релевантны каждому заданному запросу, т. е. каждому документу массива можно приписать оценку его релевантности по отношению к каждому запросу. Эти оценки делаются либо авторами запросов, либо специалистами, знакомыми с данной предметной областью.

Таблица 1

Основные характеристики массивов, использовавшихся в экспериментах по сравнению систем SMART и MEDLARS

Вид массива	Число		Источник	Тип оценки релевантности
	запросов	документов		
Исходный массив, взятый из системы MEDLARS	18	273	Часть массива документов, ранее использовавшегося в системе MEDLARS для ее оценки разработчиками системы	Оценка давалась авторами запросов для части выданных документов
Массив под названием «расширенный MEDLARS»	29	450	Получен независимо от обеих систем с помощью указателя цитированной литературы	Полная оценка каждого документа, данная внешним экспертом
Массив по офтальмологии I	29	852	Документы взяты из 9 журналов по офтальмологии независимо от обеих систем	Полная оценка, данная внешним экспертом
Массив по офтальмологии II	17	852	Тот же, что и у предыдущего массива	Оценку давали авторы запросов для 10 выданных документов (5 из них выданы системой MEDLARS, а 5 — системой SMART)
	76	1575		

В табл. 1 приводятся все характеристики массивов документов, которые использовались в описываемых экспериментах и которые были взяты из массивов MEDLARS. Среди них первые три массива документов и три множества запросов все разные, общее число запросов равно 76, число документов — 1575. Последний массив под названием Офтальмология II совпадает с массивом Офтальмология I, за исключением различий, касающихся оценок релевантности и числа запросов, использованных в экспериментах.

В следующих разделах дается обзор первых экспериментов по сравнению систем SMART и MEDLARS, а также подробно описывается организация более широкого эксперимента, в котором используются документы из области офтальмологии [22, 23].

4.2. Первые эксперименты по оценке систем

Эксперименты по сравнению систем SMART и MEDLARS, проведенные несколько лет назад, базировались на 18 запросах и 273 документах, ранее использованных для оценки работы системы MEDLARS, осуществленной в Национальной медицинской библиотеке самими разработчиками системы [24]. Автором каждого запроса была дана оценка релевантности каждого документа из этого массива именно данному запросу, однако не была проведена оценка релевантности каждого документа по отношению к каждому запросу.

В этих экспериментах работа системы SMART состояла из следующих операций: перфорирования рефератов всех документов и построения трех различных векторов, которые должны были выражать содержание документа или запроса, причем эти векторы находились для каждого запроса и каждого документа. Различались следующие виды векторов понятий:

а) вектор словоформ с отброшенной конечной буквой s (вектор s), который содержит слова из реферата документа с приписанными им весами, причем конечное s у этих слов отбрасывается;

б) вектор основ слов, состоящий из выбранных основ слов вместе с их весами, причем эти основы получаются из исходных рефератов с помощью обычной процедуры отсечения суффиксов;

в) вектор классов тезауруса, который получается с помощью некоторого вручную построенного словаря, или тезауруса, объединяющего родственные слова или основы слов в один и тот же класс тезауруса; в результате просмотра этого словаря всем исходным словам или основам слов приписываются номера соответствующих классов тезауруса, которые и задают соответствующий вектор понятий документа.

После выполнения описанного автоматического анализа векторы запроса и документа сравниваются между собой, и для каждой пары документ-запрос вычисляется некоторый коэффициент близости. Затем для каждого запроса задается величина предела выдачи, равная числу документов, выданных системой MEDLARS в ответ на этот запрос, и подсчитываются значения полноты и точности. Средние значения полноты, полученные по 18 контрольным запросам для обеих систем, приведены во втором столбце табл. 2. Из этой таблицы видно, что в системе SMART получается увеличение полноты на 8—12% по сравнению с системой MEDLARS [1]).

Таблица 2

Результаты сравнения систем SMART и MEDLARS, полученные в первом эксперименте (результаты усреднены по 273 документам и 18 запросам, для каждого запроса предел выдачи в обеих системах один и тот же)

Используемые методы анализа	Полнота Разность в %	(КрВ) (КрЗ)	Заведомая точность Разность в %	(КрВ) (КрЗ)	Условная точность Разность в %	(КрВ) (КрЗ)
MEDLARS	0,643		0,625		0,625	
SMART						
Словоформы (слова без конечного s)	0,704	(0,3124)	0,368	**0,0007**	0,571	(0,2174)
	+9%	(0,5000)	−41%	**0,0001**	−9%	(0,3145)
Основы слов	0,718	(0,2385)	0,367	**0,0007**	0,570	(0,1906)
	+12%	(0,5000)	−41%	**0,0001**	−9%	(0,3145)
Тезаурус	0,695	(0,3611)	0,393	**0,0011**	0,611	(0,3973)
	+8%	(0,5000)	−37%	**0,0002**	−2%	(0,5000)

[1]) Во всех таблицах, отражающих результаты сравнения двух систем, отмечены статистически достоверные данные. В них отражена вероятность того, что при допущении о статистической эквивалентности соответствующих множеств значений полноты и точности для разных систем наблюдаемые при этом различия, приведенные в таблицах, были бы чисто случайными. Согласно критерию Вилкоксона [КрВ], величины разностей не играют роли, имеют значение лишь ранговые статистики этих разностей. Кроме того, критерий Вилкоксона требует, чтобы две рассматриваемые случайные величины имели бы функции распределения, принадлежащие одному семейству. Критерий знаков [КрЗ] не принимает во внимание ни величину разностей, ни их ранговые статистики, а только знаки этих разностей, при этом не делается никаких предположений о характере распределения этих случайных величин.

В любом из этих случаев, вообще говоря, предполагается, что вычисленное значение вероятности, меньшее 0,05, означает, что различие двух случайных величин (двух сравниваемых методов) статистически достоверно, а гипотеза о статистической эквивалентности множеств значений должна быть отвергнута. В тех случаях, когда различие двух методов статистически достоверно, соответствующие значения разностей набраны полужирным шрифтом.

К сожалению, в этих первых экспериментах трудно было подсчитать значение точности, поскольку мы не располагали оценками релевантности каждого документа, выданного системой SMART, по отношению к каждому запросу; соответствующие оценки имелись только для документов, выданных MEDLARS. Обычно наблюдалась такая картина: для любого запроса около 5 документов из 10 выдавались как системой SMART, так и MEDLARS. Для каждого из этих общих 5 документов имелись соответствующие оценки релевантности, но об оставшихся 5 документах, выданных SMART, но не выданных MEDLARS, ничего не было известно. Значения «заведомой точности», приведенные в третьем столбце табл. 2, получены в предположении, что все документы, выданные SMART и не имеющие оценок релевантности, считаются нерелевантными для данных запросов. В четвертом столбце приводятся значения «условной точности», которая подсчитана в предположении, что доля релевантных документов, выданных системой SMART, среди неописанных документов будет той же самой, что и их доля среди документов, имеющих оценку релевантности.

Из рассмотрения табл. 2 ясно, что небольшое увеличение полноты в системе SMART по сравнению с MEDLARS компенсируется небольшим уменьшением «условной точности», из чего можно сделать вывод о том, что эти показатели эффективности для обеих систем имеют один и тот же порядок. Этот вывод подтверждается также тем, что в каждом случае различие этих характеристик статистически недостоверно. К сожалению, при подсчете «условной точности» мы были вынуждены воспользоваться допущением, достоверность которого нельзя доказать. Поэтому был проделан ряд новых экспериментов для того, чтобы получить более подробную картину, отражающую эффективность применения различных процедур системы SMART при обработке текста.

4.3. Разбиение документов по рангам и поиск с обратной связью

Вместо лингвистического анализа, слишком сложного с точки зрения его реализации на машине, можно предложить следующие три машинные процедуры, которые, по-видимому, имеют наибольшее значение:

1) Разбиение документов по рангам, осуществленное в SMART, дает возможность потребителю получать доступ прежде всего к документам, которые считаются наиболее релевантными; это предполагает установление связи между числом выданных документов, просматриваемых потребителем, и значением соответствующего коэффициента корреляции между документом и запросом.

2) Открываются новые пути построения словарей и списков слов, необходимых для автоматического анализа содержания, благодаря возможности хранить полный массив документов и вычислять коэффициенты подобия между парами векторов, характеризующих документы.

3) По-видимому, может быть достигнуто значительное увеличение эффективности поиска благодаря итеративной стратегии поиска, когда информация о документах, выданных на предыдущем этапе поиска, служит для уточнения запроса, для формулирования нового, улучшенного запроса, который используется при следующей итерации.

Рассмотрим сначала вопрос о рангах. Обычно запрос формулируется в виде булевой формулы, что, по-видимому, не дает возможности получить оптимальные результаты поиска, потому что число выданных документов обычно зависит не от реальных потребностей специалиста, а от формулировки запроса, которая мало контролируется или совсем не контролируется потребителем. Так, вполне возможна ситуация, когда для некоторых запросов выдается большое количество документов, в то время как для других не выдается ничего или выдается очень мало документов. Возможно, что, ограничивая объем выдачи в системе SMART рамками MEDLARS, как это имеет место, когда мы ограничиваем число выданных системой SMART документов числом документов, выданных в ответ на этот запрос системой MEDLARS, мы пользуемся неразумной стратегией поиска, которая приводит к худшим результатам, чем это ожидалось вначале.

Более разумной стратегией для системы SMART было бы использование корреляционного предела выдачи, при этом выдавались бы все те документы, для которых коэффициент корреляции между документом и данным запросом превосходил бы некоторую заданную величину. Тогда для того, чтобы можно было сравнивать между собой эти две системы, нужно выбрать коэффициент корреляции таким образом, чтобы общее число документов, выданных системой SMART в ответ на все запросы, совпадало с соответствующим общим числом для системы MEDLARS. Однако можно сделать так, чтобы число документов, выданное на отдельный запрос, могло меняться в соответствии с величиной коэффициента близости запрос-документ.

Существенную роль процедура ранжирования документов играет также в реализации на машине процесса обратной связи с учетом релевантности, когда в системе SMART автоматически получаются улучшенные формулировки запросов [25, 26]. Первый сеанс поиска проводится с помощью исходного заданного запроса. Затем для некоторых документов, выданных в начале этого сеанса, потребитель дает оценку релевантности, и машина

строит новый запрос, более похожий на релевантные документы и менее похожий на нерелевантные документы, чем исходный запрос. Точнее, запрос с учетом обратной связи получается путем добавления к исходному запросу терминов из тех документов, про которые известно, что они релевантны, или путем увеличения весов таких терминов; термины же, встречающиеся в нерелевантных документах, получают малый вес или вообще исключаются из запроса.

Формула, с помощью которой получается улучшенный запрос и которая на самом деле использовалась в описываемых экспериментах, может быть представлена в следующем виде:

$$q_{i+1} = q_i + \sum_{i}^{10} r_i - \sum_{j}^{2} s_j, \qquad (3)$$

где q_i — i-я итерация запроса, а каждый r_i — это i-й релевантный документ среди первых документов, число которых задается пределом выдачи; s_j обозначает один из двух нерелевантных документов среди выданных документов. Таким образом, в процессе обратной связи участвуют до 10 релевантных документов и не более 2 нерелевантных, причем предполагается, что все они находятся в множестве документов, задаваемых пределом выдачи. Если это множество содержит меньше 12 документов, тогда наибольшие значения для i и j должны быть соответственно уменьшены.

Очевидно, что в принципе можно выполнить сколько угодно итераций. На практике по мере увеличения числа итераций получаемая при этом выгода в виде увеличения числа релевантных документов среди выданных будет уменьшаться, поэтому обычно будет достаточно двух или трех итераций.

Можно сравнить операции, осуществляющие в системе SMART поиск с обратной связью, с работой некоторой программы анализа текста, в том смысле, что получение улучшенного запроса эквивалентно повторной индексации документов или формулированию заново содержания запроса. В обычных ИПС теоретически возможно использовать поиск с обратной связью — например, попросить потребителя заново сформулировать запрос, пользуясь информацией, выданной ему на предыдущем этапе; на практике же не так-то легко осуществить эту идею. Одна из непосредственно возникающих трудностей состоит в том, что почти всюду используется инвертированный способ организации поискового массива, при котором компоненты вектора понятий данного документа разбросаны по разным частям этого массива. При таких условиях для того, чтобы получить вектор документа целиком, нужно иметь многоканальный доступ к массиву, что затрудняет практическое использование операций типа поиска с обратной связью.

4.4. Автоматическое построение словарей

Составление тезауруса, который можно использовать для стандартизации языка документов, можно разбить на два этапа. Во-первых, нужно выбрать те термины, которые действительно используются при описании содержания документов. Во-вторых, выбранные слова, отражающие содержание документа, нужно объединить в классы эквивалентности так, чтобы все термины, включенные в некоторый данный класс тезауруса, были представлены в векторе документа некоторым одним общим идентификатором. На первом этапе пытаются исключить из описаний содержания документов все те термины, которые на самом деле не отражают сути документа, или такие термины, которые не способствуют выделению данного документа среди всех документов массива, — обычно это бывают функциональные слова, такие, как артикли, предлоги, союзы и т. п., а также слова, употребляемые с большой частотой в рассматриваемой предметной области. Описанные ниже процедуры позволяют объединять в некоторые классы синонимичные слова или слова, тесно связанные между собой каким-то другим образом. Таким образом, эквивалентность этих терминов обнаруживается при некотором автоматическом анализе текста.

Существует много алгоритмов автоматической классификации терминов [13, 14, 27—31], причем некоторые из них дают возможность получить такую классификацию терминов, которая при информационном поиске не уступает классификации, полученной вручную. Главным препятствием на пути развития автоматических методов классификации служит относительно высокая стоимость затрат времени и труда, которые требуются для построения автоматического тезауруса. Во многих случаях приходится разбивать на классы несколько десятков тысяч терминов, а число сравнений векторов из n терминов имеет порядок n^2 или по меньшей мере kn, т. е. слишком велико для практического применения. Поэтому прежде, чем эти методы будут свободно применяться на практике, нужно разработать более дешевые методы автоматической классификации [32].

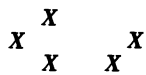
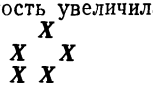
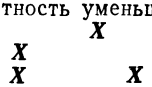
В то время как проблема классификации еще ждет своего решения, нужные словари могут быть построены без проведения какой бы то ни было классификации путем простого внесения в словарь таких терминов, которые считаются полезными при анализе содержания, или путем отбрасывания неинформативных терминов; иногда их называют общеупотребительными словами. Обычно выделение общеупотребительных слов является довольно сложной операцией, при которой необходим анализ семантических свойств отдельных слов, а также способов употребления этих слов в документах данного массива. При таких обстоятель-

ствах решение вопроса, использовать ли данное слово для описания содержания или нет, достаточно часто должно приниматься интуитивно.

Недавно был осуществлен ряд экспериментов, целью которых было нащупывание путей к тому, чтобы превратить сложный процесс отбора общеупотребительных слов в автоматический, управляемый процесс [33, 34]. В основе этих экспериментов лежит следующее наблюдение: если в качестве идентификатора содержания документа взято общеупотребительное слово, соответствующий термин будет встречаться во многих документах, делая их таким образом похожими друг на друга, и наоборот, если документу приписывается некоторый идентификатор, действительно отражающий содержание именно этого документа,

Таблица 3

Изменения, происходящие в пространстве документов

Документы в исходный момент	Документы после того, как к вектору документа добавлены слова-неразличители (общеупотребительные слова)	Документы после того, как к векторам документов добавлены слова-различители (значимые слова)
1. Пространство документов		
Пространство в исходный момент	Плотность увеличилась	Плотность уменьшилась
		
2. Векторы документов D_i и D_j		
$(x_i^1, x_i^2, \ldots, x_i^k, \ldots, x_i^n, \ldots)$	$(x_i^1, x_i^2, \ldots, x_i^k, \ldots, x_i^n, x_i^{n+1}, x_{i,\ldots}^{n+2})$	$(x_i^1, x_i^2, \ldots, x_i^k, \ldots, x_i^n, x_i^{n+1}, x_i^{n+2}, \ldots)$
$(x_j^1, x_j^2, \ldots, x_j^k, \ldots, x_j^n, \ldots)$.	$(x_j^1, x_j^2, \ldots, x_j^k, \ldots, x_j^n, x_j^{n+1}, x_{j,\ldots}^{n+2})$	$(x_j^1, x_j^2, \ldots, x_j^k, \ldots, x_j^n, x_j^{n+1}, x_j^{n+2}, \ldots)$
3. Вычисление коэффициента близости		
(3 одинаковых термина из 10)	(5 одинаковых терминов из 12)	(3 одинаковых термина из 12)
$S(D_i, D_j) = \frac{3}{10+10-3} = 0{,}1764$	$S(D_i, D_j) = \frac{5}{12+12-5} = 0{,}2631$	$S(D_i, D_j) = \frac{3}{12+12-3} = 0{,}1428$

то это слово позволит выделить данный документ среди других, при этом векторы отдельных документов имеют меньшую близость.

Это положение иллюстрируется примером, приведенным в табл. 3. В верхней части этой таблицы каждый документ изображается в пространстве документов в виде X, а расстояние между двумя X обратно пропорционально величине близости соответствующих векторов документов. Левая часть табл. 3 изображает исходную ситуацию. Для примера предполагается, что документы D_i и D_j имеют три общих термина. Тогда коэффициент корреляции, измеряемый числом общих терминов, равен 0,1764. В центре таблицы представлена ситуация, когда к векторам документов D_i и D_j добавлено по два общеупотребительных слова, в результате чего эти документы теперь имеют 5 общих терминов и коэффициент корреляции, равный 0,2631. Очевидно, что теперь пространство документов как бы сжалось. В правой части табл. 3 к векторам документов D_i и D_j добавляется по два информативных слова, причем маловероятно, чтобы они оказались общими для D_i и D_j. Теперь эти два документа пересекаются по 3 терминам из 12, коэффициент корреляции равен 0,1428, а пространство документов расширилось по сравнению с исходным.

Теперь ясна стратегия, которой следует придерживаться, чтобы отличить общеупотребительные слова от информативных, или значимых терминов. А именно вычисляется некая функция, представляющая плотность Q пространства документов (например, Q может представлять собой сумму корреляций между каждым документом и центром пространства). В пространство документов вводится новый термин i; если новое значение функции плотности пространства Q_i больше Q, то i — неинформативное слово, или слово-неразличитель, а разность $Q_i - Q$ выражает силу этого слова как слова-неразличителя. Наоборот, если в случае добавления нового термина i пространство расширяется и Q_i меньше Q, то термин i — значащее слово, а разность $Q - Q_i$ измеряет его силу как значащего слова, или слова-различителя.

Располагая слова-различители в порядке уменьшения разности $Q - Q_i$, мы получаем список, в котором самые сильные слова-различители стоят на первых местах. Аналогично, располагая слова-неразличители в порядке уменьшения разности $Q_i - Q$, в начале списка мы получаем слова, наименее полезные для выделения любого документа среди других документов. В табл. 4 приведены два таких списка, содержащих по 12 самых сильных различителей и неразличителей из области офтальмологии. Списки получены на массиве из 852 документов.

Таблица 4

Список слов-различителей и неразличителей, расположенных в порядке убывания их различительной (неразличительной) силы

Неразличители				Различители			
Слово	Число документов, в которых встретилось данное слово	Суммарная частота появления данного слова	Средняя частота	Слово	Число документов, в которых встретилось данное слово	Суммарная частота появления данного слова	Средняя частота
1. Patient	201	408	2,03	1. Rubella	10	47	4,70
2. At	194	292	1,51	2. Capillary	19	54	2,84
3. Use	179	247	1,38	3. Laser	11	32	2,91
4. Have	194	257	1,32	4. Collagen	12	40	3,33
5. Retinal	134	275	2,05	5. Cyst	17	42	2,47
6. Present	184	219	1,19	6. Cholinesterase	6	26	4,33
7. Has	171	231	1,35	7. Fiber	16	50	3,13
8. Effect	150	259	1,73	8. Cyclodialysis	4	12	3,00
9. Result	179	234	1,31	9. Implant	18	36	2,00
10. Found	174	228	1,31	10. Uveitis	21	45	2,14
11. Report	141	172	1,22	11. Vessel	36	82	2,28
12. Occular	125	194	1,55	12. Spray	2	25	12,50

Таким образом, для того чтобы построить автоматически словарь значащих слов, или слов-различителей, нужно вычислить значение функции плотности пространства документов для каждого термина из массива документов. Таблица 5 иллюстрирует процесс построения такого словаря для массива документов по офтальмологии.

Прежде всего из 852 рефератов, содержащихся в массиве по офтальмологии, выбираются все различные слова. После того как у слов, оканчивающихся на s, удалена последняя буква, мы получаем так называемый словарь «словоформ». Затем из этого списка вычеркиваются слова, встретившиеся по одному разу во всем массиве, а также слова, встретившиеся в 25% документов или более того, поскольку довольно ясно, что термины с частотой 1 не будут иметь значения при сравнении документа и запроса, а термины, встречающиеся с большой частотой, не позволяют выделить из массива нужные документы. К 5100 оставшимся терминам применяется алгоритм выделения слов-различителей, в результате чего около 200 терминов вычеркиваются как слова-неразличители. Как показано в последней строке табл. 5, можно удалить дополнительно еще несколько терминов,

если начать с удаления наиболее слабых слов-различителей и вычеркивать слова в порядке увеличения их силы как различителей.

Таблица 5

Построение при помощи машины некоторых списков слов (на базе массива по офтальмологии)

Процедура, применяемая для построения словаря	Вид получающегося словаря	Число единиц в словаре	
		удаленных из словаря	оставшихся
Составление списка слов, встретившихся в рефератах документов, с последующим отбрасыванием конечной буквы s	Словарь «s» (словоформы без конечного «s»)	—	8672
Вычеркивание терминов с частотой употребления, равной 1	—	3535	5137
Вычеркивание терминов, встретившихся в 25% и более от числа всех документов	Автоматический список слов	29	5108
Вычеркивание терминов, определенных машиной как слова-неразличители (минимум плотности пространства документов)	Автоматический словарь слов-различителей	180	4928
Дополнительное вычеркивание терминов в порядке уменьшения их различительной способности	Сокращенный словарь слов-различителей	3928	1000

В следующем разделе представлены результаты поиска, изучение которых позволяет сравнить эффективность применения различных автоматических процедур, реализованных в системе SMART, — ограничения числа выдаваемых документов с помощью корреляционного предела выдачи, использования автоматических словарей слов-различителей и поиска с обратной связью — с результатами работы системы MEDLARS.

4.5. Общие результаты

В эксперименте использовались два массива документов из области медицины, причем эти массивы не зависят ни от системы SMART, ни от системы MEDLARS. Массив под названием

«расширенный MEDLARS» состоит из 450 документов, названия которых были выбраны из Указателя цитированной литературы по точным, естественным и прикладным наукам (Science Citation Index), причем ссылки на эти документы имелись в ряде документов, отмеченных как релевантные некоторым запросам потребителей системы MEDLARS [23]. Массив же документов по офтальмологии состоит из статей, опубликованных в 9 основных журналах по офтальмологии в 1967 году[1]). Для обоих этих массивов была проведена проверка, введен ли каждый документ в действующую систему MEDLARS. Если некоторый документ не введен в эту систему, то он должен быть исключен из массива до начала эксперимента по сравнению систем.

В эксперименте использовалось 58 запросов, ранее заданных системе MEDLARS ее потребителями, из них 29 запросов по офтальмологии и 29 запросов по биомедицинской тематике. Если авторами запросов являются ученые и врачи, как это имеет место для большинства запросов, поступающих в систему MEDLARS, практически невозможно требовать от них полной оценки релевантности, т. е. оценки релевантности каждого документа каждому запросу. Поэтому мы были вынуждены воспользоваться полной оценкой релевантности, данной внешними экспертами; для массива «расширенный MEDLARS» эта оценка была дана студентом-медиком, а для массива по офтальмологии — специалистом в этой области. В табл. 6 приводятся усредненные результаты поиска по массиву «расширенный MEDLARS» в условиях внешней экспертной оценки, а в табл. 7 — аналогичные результаты для массива по офтальмологии. Из рассмотрения этих таблиц можно сделать следующие основные выводы:

1) В том случае, когда число документов, выдаваемых в ответ на запрос, для обеих систем одинаково, алгоритмы автоматического анализа текста, работающие в системе SMART, приводят к потере полноты и точности, меняющейся от 20 до 50%, по сравнению с результатами системы MEDLARS, где применяется обычное индексирование документов. При этом наблюдаемые различия в значениях полноты и точности статистически достоверны, так что ясно, что система индексирования MEDLARS дает лучшие результаты.

2) В том случае, когда в системе SMART с помощью корреляционного предела выдача организуется так, чтобы общее число выданных документов в ответ на все 29 запросов было одинаковым для обеих систем и составляло 127 документов для

---

[1]) Acta Ophthalmologica, American Journal of Ophthalmology, Archives of Ophthalmology, British Journal of Ophthalmology, Experimental Eye Research, Investigative Ophthalmology, Ophthalmologica, Transactions of the American Academy of Ophthalmology and Otolaryngology и Vision Research.

Таблица 6

Результаты сравнения систем SMART и MEDLARS на массиве «расширенный MEDLARS» (450 документов, 29 запросов, причем в системе SMART использовался корреляционный предел задачи, так что число всех выданных документов равнялось 127)

Используемые средства анализа	Предел выдачи, равный пределу выдачи в системе MEDLARS		Корреляционный предел выдачи		Две итерации при поиске с обратной связью	
	Полнота (КрВ) Разность (КрЗ) в %	Точность (КрВ) Разность (КрЗ) в %	Полнота (КрВ) Разность (КрЗ) в %	Точность (КрВ) Разность (КрЗ) в %	Полнота (КрВ) Разность (КрЗ) в %	Точность (КрВ) Разность (КрЗ) в %
Система MEDLARS (контролируемое индексирование)	0,3117	0,6110				
Словарь «s»	0,1814 **0,0021** −42% **0,0005**	0,3867 **0,0007** −37% **0,0021**	0,2613 (0,2216) −16% (0,3450)	0,4960 (0,1302) −19% (0,1050)	0,3525 (0,3383) +13% (0,5000)	0,6740 (0,2665) +13% (0,5000)
Словарь основ слов	0,1814 **0,0013** −42% **0,0005**	0,4141 **0,0009** −32% **0,0005**	0,2622 (0,2420) −16% (0,4194)	0,4901 **0,0494** −19% **0,0392**	0,3433 (0,3939) +10% (0,5000)	0,6892 (0,2021) +13% (0,3318)
Автоматический словарь слов-различителей A5	0,2462 **0,0051** −21% **0,0154**	0,4518 **0,0032** −26% **0,0154**	0,2872 (0,3102) −8% (0,2781)	0,5879 (0,4056) −4% (0,2706)	0,3801 (0,1870) +22% (0,4223)	0,7230 (0,1216) +18% (0,3238)
Тезаурус	0,2181 **0,0000** −30% **0,0032**	0,4512 **0,0018** −26% **0,0032**	0,3232 (0,4946) +4% (0,3450)	0,6106 (0,3102) 0% (0,4223)	0,4029 (0,1578) +29% (0,0849)	0,7438 (0,0914) +22% (0,3450)

Таблица 7

Результаты сравнения систем SMART и MEDLARS на массиве по офтальмологии (852 документа, 29 запросов, оценка документов дана внешними экспертами)

Средства анализа	Предел выдачи равен пределу выдачи в системе MEDLARS		Одна итерация при поиске с обратной связью		Две итерации при поиске с обратной связью [1]	
	Полнота (КрВ) Разность (КрЗ) в %	Точность (КрВ) Разность (КрЗ) в %	Полнота (КрВ) Разность (КрЗ) в %	Точность (КрВ) Разность (КрЗ) в %	Полнота (КрВ) Разность (КрЗ) в %	Точность (КрВ) Разность (КрЗ) в %
Система MEDLARS (контролируемое индексирование)	0,4272	0,4454				
Система SMART						
Словарь «s»	0,2240 **0,0007** −48% **0,0036**	0,2728 **0,0023** −39% **0,0036**	0,4025 (0,4861) −6% (0,5000)	0,4144 (0,1239) −7% (0,0155)	0,4181 (0,3975) −2% (0,4159)	0,4839 (0,3658) +9% (0,2706)
Словарь основ слов	0,2802 **0,0056** −34% **0,0207**	0,2932 **0,0089** −34% **0,0207**	0,4125 (0,4741) −3% (0,4159)	0,4367 (0,2505) −2% (0,0388)	0,4318 (0,3365) +1% (0,4159)	0,4402 (0,1379) −1% (0,0539)
Автоматический список слов А4	0,2843 **0,0134** −33% **0,0318**	0,3156 **0,0109** −29% **0,0318**	0,4251 (0,4310) 0% (0,5000)	0,5249 (0,2821) +18% (0,5000)	0,4440 (0,3195) +4% (0,5000)	0,6055 (0,0892) +36% (0,5000)
Автоматический словарь слов-различителей А5	0,3159 **0,0125** −26% **0,0207**	0,3039 **0,0072** −32% **0,0207**	0,4624 (0,2374) +8% (0,1050)	0,4041 (0,1480) −9% (0,5000)	0,4794 (0,1650) +12% (0,1050)	0,4456 (0,4144) 0% (0,2207)
Тезаурус	0,3355 **0,0149** −21% **0,0318**	0,3262 **0,0182** −26% **0,0318**	0,4230 (0,4851) −1% (0,2517)	0,4403 (0,3102) −1% (0,4223)	0,4475 (0,3074) +5% (0,0669)	0,4727 (0,4445) +6% (0,4223)

[1]) Поиск с обратной связью, выполняемый в системе SMART, осуществляется с помощью корреляционного предела выдачи, заданного таким образом, чтобы выдавалось 602 документа.

массива «расширенный MEDLARS» и 602 документа для массива по офтальмологии, тогда потери в полноте и точности, которые дает система SMART, уменьшаются примерно на 10%, причем эти различия не являются теперь статистически достоверными. Это означает, что, рассматривая полученные результаты, нельзя сделать достоверного вывода о преимуществе той или иной системы.

3) Поиск с обратной связью, реализованный в системе SMART, дает увеличение эффективности по сравнению с MEDLARS, колеблющееся от нескольких процентов для одной итерации при поиске до 30 процентов для двух итераций, при этом в большинстве случаев различия соответствующих характеристик эффективности статистически недостоверны.

4) При поиске с обратной связью основной алгоритм, используемый в системе SMART, который дает возможность приписать документам и запросам взвешенные основы слов, уже может соперничать с ручными методами индексирования, применяемыми в MEDLARS. Это означает, что можно успешно применять некоторую процедуру извлечения из текста основ слов, дополненную соответствующими изменениями в процессе поиска.

5) В том случае, когда в системе SMART используется автоматический словарь слов-различителей, а также при использовании вручную построенного тезауруса получаемые при этом результаты вполне сравнимы с результатами, найденными в MEDLARS, причем эти результаты получаются без учета обратной связи, и для более удобного управления объемом выдачи применяется только ранжирование документов. Если же нормализацию языка, обеспечиваемую автоматическими словарями, дополнить поиском с обратной связью, то можно достичь увеличения эффективности по сравнению с системой MEDLARS до 30%.

Главный вывод, сделанный в результате рассмотрения табл. 6 и 7, можно сформулировать следующим образом.

Маловероятно, чтобы обыкновенный выбор слов из реферата или текста документа с последующим поиском по запросу, задаваемому в виде булевой формулы, который применяется сейчас во многих ИПС, был бы столь же эффективен, что и обычное индексирование документов, выполняемое человеком. Однако можно легко заставить машину выполнять другие, вообще говоря нелингвистические операции, такие, как ранжирование документов, нормализация текста с помощью машинных словарей и тезаурусов, итеративный поиск, благодаря которым можно будет достичь большей эффективности, чем та, которую дают системы, контролируемые человеком.

Следует отметить, что абсолютные значения цифр, приведенных в табл. 6 и 7, не являются точными значениями ни одного

конкретного показателя эффективности, но разности этих величин для двух или более методов или систем действительно отражают относительные уровни эффективности. Это происходит потому, что существуют некоторые предельные значения полноты и точности, выше которых эти характеристики не поднимаются.

Таблица 8

Предельные значения полноты и точности для массива по офтальмологии (внешняя экспертная оценка релевантности, 852 документа, 29 запросов)

Номер запроса	Число документов, выданных системой MEDLARS	Число документов в массиве, релевантных данному запросу	Верхний предел для числа выданных релевантных документов	Предельное значение полноты	Предельное значение точности
1	20	14	14	1,0000	0,7000
2	11	1	1	1,0000	0,0909
3	5	6	5	0,8333	1,0000
4	36	92	36	0,3913	1,0000
5	8	60	8	0,1333	1,0000
6	16	10	10	1,0000	0,6250
7	54	59	54	0,9153	1,0000
8	6	2	2	1,0000	0,3333
9	19	15	15	1,0000	0,7895
10	8	8	8	1,0000	1,0000
11	37	5	5	1,0000	0,1351
12	12	23	12	0,5217	1,0000
13					
14	5	1	1	1,0000	0,2000
15	7	53	7	0,1321	1,0000
16	28	24	24	1,0000	0,8571
17	20	1	1	1,0000	0,0500
18					
19	17	14	14	1,0000	0,8235
20	11	54	11	0,2037	1,0000
21	12	7	7	1,0000	0,5833
22	10	54	10	0,1852	1,0000
23	6	65	6	0,0923	1,0000
24	10	43	10	0,2326	1,0000
25	10	33	10	0,3030	1,0000
26	11	12	11	0,9167	1,0000
27	5	1	1	1,0000	0,2000
28	174	149	149	1,0000	0,8563
29	8	1	1	1,0000	0,1250
30					
31	6	4	4	1,0000	0,6667
32					
33					
34	30	44	30	0,6818	1,0000
Среднее предельное значение				0,7428	0,7254

Пределы же полноты и точности возникают потому, что число документов, выданных в ответ на данный запрос, как правило, не совпадает с числом релевантных документов, определенным для этого запроса с помощью внешней экспертной оценки. Данные табл. 8 говорят о том, что из 852 документов по офтальмологии только 602 были оценены как релевантные 29 рассматриваемым запросам. В случае когда величина, определяемая как число документов, релевантных данному запросу, меньше предела выдачи в системе MEDLARS, как это имеет место для запроса под номером 1 в табл. 8, точность с необходимостью принимает значения меньше 1. Наоборот, когда число релевантных документов больше числа выданных, как в случае запросов с номерами 3, 4 и 5, тогда ограничения налагаются на значения полноты. Для массива по офтальмологии среднее значение точности и полноты заключено между 70 и 75%. Таким образом, самые высокие результаты, содержащиеся в табл. 7, составляют 65% максимального значения полноты и 83% максимального возможного значения точности.

4.6. Эксперименты, опирающиеся на избирательные оценки релевантности, данные авторами запросов

В предыдущем разделе были приведены результаты поиска, оцененные не авторами запросов, а специалистами в данной предметной области, которые приписывали каждому документу полную оценку релевантности. Если такую оценку дают непосредственно авторы запросов, то, как правило, эта информация бывает неполной, поэтому необходимо как-то приспособить к этим условиям методы оценки результатов поиска. Кроме того, если оценку релевантности имеют лишь некоторые документы, могут возникнуть трудности при интерпретации результатов, о которых уже говорилось при рассмотрении табл. 2.

Тогда была сделана попытка преодолеть эти трудности следующим образом. Каждому из авторов 29 запросов по офтальмологии было предложено оценить 10 документов, выданных ранее в ответ на его запрос, при этом 5 документов были получены случайной выборкой из документов, выданных системой MEDLARS, а 5 документов — случайной выборкой из документов, выданных системой SMART с обычным использованием основ слов. Каждого автора попросили оценить все 10 документов с точки зрения релевантности запросу данного автора. При этом разрешалось давать только три оценки: документ имеет важное значение для автора запроса, небольшое значение и не имеет никакого значения. Результаты авторских оценок сводились в таблицы, причем рассматривалось два случая: 1) релевантными считаются документы, имеющие хоть какое-нибудь

Таблица 9

Сравнение значений относительной полноты для массива по офтальмологии (усреднение по 17 запросам оценок релевантности, данных авторами запросов, для 5 документов, выданных системой SMART, и 5 документов, выданных MEDLARS)

Средства анализа	Словоформы		Основы слов		Автоматический список слов		Список слов-различителей		Тезаурус	
	Полнота %	(КрВ) (КрЗ)	Полнота %	(КрВ) (КрЗ)	Полнота %	(КрВ) (КрЗ)	Полнота %	(КрВ) (КрЗ)	Полнота %	(КрВ) (КрЗ)
(1) Предел выдачи в системе SMART совпадает с пределом выдачи в системе MEDLARS										
MEDLARS	0,2490	(0,3283)	0,2294	(0,4070)	0,1931	(0,1870)	0,1274	(0,2871)	0,2568	(0,2108)
SMART	0,2774	(0,2744)	0,2509	(0,3872)	0,2794	(0,2744)	0,1696	(0,1445)	0,3068	(0,5000)
	+11%		+9%		+45%		+33%		+19%	
(2) Корреляционный предел выдачи										
MEDLARS	0,1666	(0,0486)	0,2107	(0,2386)	0,1588	(0,1540)	0,1372	(0,1013)	0,1294	**0,0092**
SMART	0,2941	(0,0195)	0,2676	(0,2539)	0,2588	(0,1719)	0,2735	(0,1719)	0,3264	**0,0327**
	+76%		+27%		+61%		+100%		+151%	
(3) Одна итерация при поиске с обратной связью										
MEDLARS	0,1666	(0,1432)	0,1519	(0,1871)	(0,1588)	**0,0333**	0,1078	**0,0515**	0,1294	**0,0017**
SMART	0,2372	(0,2539)	0,1960	(0,2539)	0,2754	**0,0547**	0,2696	**0,0547**	0,3254	**0,0107**
	+42%		+29%		+73%		+150%		+151%	
(4) Две итерации										
MEDLARS	0,1960	(0,0712)	0,1519	(0,0618)	0,1588	**0,0333**	0,1078	**0,0515**	0,1294	**0,0047**
SMART	0,2754	(0,2744)	0,1960	(0,2539)	0,2754	**0,0547**	0,2696	**0,0547**	0,3254	**0,0107**
	+41%		+29%		+73%		+150%		+151%	

Таблица 10

Сравнение значений точности для массива по офтальмологии (усреднение по 17 запросам оценок релевантности, данных авторами запросов для 10 выданных документов)

Используемые средства анализа	Словоформы Точность (КрВ) (КрЗ)	Основы слов Точность (КрВ) (КрЗ)	Автоматический список слов Точность (КрВ) (КрЗ)	Список слов-различителей Точность (КрВ) (КрЗ)	Тезаурус Точность (КрВ) (КрЗ)
(1) Предел выдачи совпадает с пределом выдачи в системе MEDLARS Точность системы MEDLARS равна 0,4823	0,3523 **0,0067** —27% **0,0037**	0,3705 **0,0240** —23% **0,0287**	0,3235 **0,0087** —33% **0,0017**	0,2000 **0,0002** —59% **0,0000**	0,3117 **0,0024** —35% **0,0021**
(2) Корреляционный предел выдачи Точность системы MEDLARS равна 0,4823	0,2764 **0,0034** —43% **0,0032**	0,2882 **0,0039** —40% **0,0112**	0,3058 **0,0093** —37% **0,0032**	0,2352 **0,0007** —51% **0,0001**	0,2823 **0,0032** —41% **0,0009**
(3) Одна итерация при поиске с обратной связью Точность системы MEDLARS равна 0,4823	0,2705 **0,0014** —44% **0,0065**	0,2705 **0,0009** —44% **0,0037**	0,3294 **0,0109** —32% **0,0287**	0,2588 **0,0009** —46% **0,0017**	0,2882 **0,0032** —40% **0,0009**

значение для автора запроса. Пригодные для использования оценки были получены для 17 из 29 запросов. Таким образом, в табл. 9, 10 и 11 приведены значения характеристик эффективности, подсчитанные при выдаче 10 документов из 852 в ответ на каждый из 17 запросов.

Если оценка релевантности документов имеется не для всех документов, нельзя подсчитать значения полноты и точности, пользуясь формулами (1) и (2). Вместо этого нужно выводить значение точности, рассматривая лишь 10 документов для каждого запроса, как это имеет место в нашем случае, а для полноты приходится давать новое определение. В нашем эксперименте точность вычислялась по следующим формулам:

$$\text{Точность в системе MEDLARS} = \frac{\text{Общее число релевантных документов, выданных MEDLARS (из 10, имеющих оценку релевантности)}}{\text{Общее число выданных документов, имеющих оценку релевантности (равное 10)}}, \quad (4)$$

$$\text{Точность в системе SMART} = \frac{\text{Число релевантных документов, выданных SMART (из 10, имеющих оценку релевантности)}}{\text{Общее число выданных документов, имеющих оценку релевантности (равное 10)}}. \quad (5)$$

Если для вычисления значения точности можно использовать довольно обычную процедуру, значение полноты получить гораздо труднее, поскольку нам неизвестно число всех документов массива, которые были бы отмечены как релевантные автором запроса. Поэтому вводится понятие относительной полноты [1]), определяемое как отношение числа релевантных документов, выданных системой MEDLARS, к числу релевантных документов, выданных системой SMART, или можно рассматривать обратное отношение. Пусть из 5 документов, выданных MEDLARS и имеющих данную ранее оценку релевантности, M документов были отмечены теперь как релевантные автором некоторого запроса. Пусть S_M документов из этих M документов выданы системой SMART, тогда относительная полнота системы SMART определяется как отношение $\frac{S_M}{M}$. Аналогично рассмотрим 5 документов, выданных SMART и имеющих оценку релевантности. Пусть S из этих документов определены как релевантные, причем M_S из этих S документов выданы системой

[1]) Предложенное М. Э. Леском из Лабораторий компании Белл.

Таблица 11

Относительная полнота и точность для массива по офтальмологии (оценки релевантности, данные авторами запросов, усреднены по 17 запросам, при этом релевантными документами считаются документы, имеющие важное значение)

Средства анализа	Словоформы		Основы слов		Автоматический список слов		Автоматический список слов-различителей		Тезаурус	
(1) Предел выдачи совпадает с пределом выдачи в MEDLARS										
Полнота: MEDLARS	0,1245	**0,0697**	0,1294	(0,1311)	0,1470	(0,1206)	0,0784	(0,1473)	0,1666	(0,0865)
SMART	0,2784	**0,0547**	0,2098	(0,3769)	0,2705	(0,3770)	0,1607	(0,1094)	0,3176	(0,0898)
%	+124%		+62%		+84%		+105%		+91%	
Точность: MEDLARS	0,2588	**0,0442**		**0,0119**		**0,0011**				**0,0060**
SMART	0,1705	**0,0176**	0,1529	**0,0065**	0,1411	**0,0002**	0,1000	**0,0011**	0,1588	**0,0032**
%	−34%		−41%		−45%		−61%	**0,0005**	−39%	
(2) Корреляционный предел выдачи										
Полнота: MEDLARS	0,0539	**0,0086**	0,0588	(0,0212)	0,1127	**0,0963**	0,0882	(0,0253)	0,1176	**0,0332**
SMART	0,3333	**0,0351**	0,2843	(0,1445)	0,2333	**0,2539**	0,3186	(0,0898)	0,3235	**0,0195**
%	+518%		+384%		+107%		+261%		+175%	
Точность: MEDLARS	0,2588	**0,0093**		**0,0017**		**0,0040**		**0,0011**		**0,0054**
SMART	0,1411	**0,0032**	0,1176	**0,0005**	0,1352	**0,0037**	0,1352	**0,0002**	0,1529	**0,0032**
%	−45%		−23%		−48%		−48%		−41%	
(3) Одна итерация при поиске с обратной связью										
Полнота: MEDLARS	0,0539	**0,0086**	0,0489	**0,0086**	0,0930	**0,0076**	0,0588	**0,0122**	0,1176	**0,0332**
SMART	0,3627	**0,0352**	0,3264	**0,0352**	0,3509	**0,0195**	0,2794	**0,0195**	0,3431	**0,0195**
%	+573%		+567%		+277%		+375%		+192%	
Точность: MEDLARS	0,2588	**0,0115**		**0,0054**		**0,0192**		**0,0011**		**0,0072**
SMART	0,1470	**0,0032**	0,1156	**0,0017**	0,1705	**0,0065**	0,1352	**0,0002**	0,1588	**0,0059**
%	−43%		−55%		−34%		−48%		−39%	

MEDLARS. Тогда относительная полнота системы MEDLARS определяется как отношение $\frac{M_S}{S}$[1]. Другими словами,

$$\text{Относительная полнота системы SMART} = \frac{\text{Число релевантных документов, выданных SMART, из общего числа релевантных документов, ранее выданных системой MEDLARS}}{} = \frac{S_M}{M}, \quad (6)$$

$$\text{Относительная полнота системы MEDLARS} = \frac{\text{Число релевантных документов, выданных системой MEDLARS, из общего числа релевантных документов, ранее выданных системой SMART}}{} = \frac{M_S}{S}. \quad (7)$$

Поскольку относительная полнота системы MEDLARS зависит от числа релевантных документов, выданных SMART, используя различные методы поиска в системе SMART, мы будем получать разные значения относительной полноты для системы MEDLARS.

В табл. 9 приведены значения относительной полноты, при этом релевантными считаются документы, имеющие для автора запроса хоть какое-нибудь значение. Рассматривая эту таблицу, можно сказать, что в том случае, когда размеры выдачи определяются пределом выдачи в MEDLARS, относительная полнота системы SMART в среднем на 25% выше относительной полноты системы MEDLARS. Если же использовать более тонкие методы, разработанные в SMART, такие, как корреляционный предел выдачи и итеративный поиск, то SMART дает улучшение относительных характеристик эффективности в среднем на 80—90% по сравнению с системой MEDLARS, при этом различие статистически достоверно. При этом наилучшие результаты получаются в том случае, когда в SMART применяются различные машинные словари, а итеративный поиск в этом случае не дает значительного улучшения. Причина этого состоит, конечно, в том, что релевантные документы, на основе которых строится улучшенный запрос, должны входить в множество из 5 документов, имеющих оценку релевантности. К сожалению, последние 5 документов выбирались довольно случайно, т. е. не были

[1]) Те запросы, для которых или M, или S равнялись нулю, были исключены из тех 17 запросов, результаты обработки которых имеются в табл. 9—11.

первыми пятью документами в выдаче. Поэтому ясно, что поиск с обратной связью в системе SMART, который по существу использует ранжированную выдачу документов, не будет давать никаких преимуществ, если для уточнения запроса брать произвольным образом упорядоченные документы.

Данные табл. 10 показывают, что значения точности, обеспечиваемые системой SMART, примерно на 30—40% ниже соответствующего значения точности (0,4823), которое получается в системе MEDLARS. Здесь опять трудность заключается в том, что при выборе документов, которые затем даются авторам запросов для оценки релевантности, не используется ранжирование документов, предусмотренное в системе SMART. В то время как преимущество в полноте для системы SMART растет от 10% при использовании списка основ слов до 150% для большинства случаев применения тезауруса, проигрыш в точности, однако, остается почти постоянным.

Как это следует из данных, приведенных в табл. 11, та же самая картина наблюдается и в случае, когда релевантными считаются документы, про которые сказано, что они имеют важное значение для автора запроса. Постоянные потери в точности для системы SMART (около 40%) компенсируются преимуществом в относительной полноте, изменяющейся от среднего значения порядка 90% для обычного предела выдачи, принятого в MEDLARS, до среднего значения в 290% для корреляционного предела выдачи и почти до 400% для первой итерации при поиске с обратной связью. Различие показателей эффективности, приведенных в табл. 11, по большей части статистически достоверно. Еще раз подтверждается, что более тонкие процедуры, используемые в SMART, позволяют увеличивать значения полноты, не вызывая при этом больших потерь в точности.

4.7. Заключение

В табл. 12 и 13 приведены усредненные разности значений показателей эффективности двух систем, причем в табл. 12 отражены разности соответствующих значений показателей для разных методов поиска, а в табл. 13 — для разных словарей, применяемых в системе SMART при поиске. Данные табл. 12 и 13 получены на основе данных таблиц 6, 7, 9, 10 и 11, при этом в табл. 12 данные усреднялись по 5 используемым словарям, а в табл. 13 — по различным методам поиска. Нам кажутся обоснованными следующие выводы:

1) По сравнению с обычным поиском по запросу, задаваемому в виде булевской формулы, значительное улучшение показателей эффективности получается с помощью ранжирования

Таблица 12

Значения разностей показателей эффективности для систем SMART и MEDLARS (усреднение по нескольким словарям, используемым в системе SMART) [1])

Процедура поиска в системе SMART	Массив «расширенный MEDLARS»		Массив по офтальмологии					
			Внешняя экспертная оценка		Оценка релевантности дана авторами запросов (релевантными считаются документы, имеющие хоть какое-нибудь значение)		Оценка релевантности дана авторами запросов (релевантными считаются документы, имеющие важное значение)	
	П	Т	П	Т	П	Т	П	Т
Предел выдачи тот же, что и в MEDLARS	−33,75%	−30,25%	−32,4%	−32,0%	+23,4%	−35,4%	+93,2%	−44,0%
Корреляционный предел выдачи	−9,00%	−10,50%	−26,2%	−26,8%	+83,0%	−42,4%	+289,0%	−41,0%
Одна итерация при поиске с обратной связью	+13,75%	+12,25%	−0,4%	−0,2%	+89,0%	−41,2%	+396,8%	−43,8%
Две итерации при поиске с обратной связью	+18,50%	+17,25%	+4,0%	+10,0%	+88,8%	−40,0%	—	—

[1]) П — полнота, Т — точность.

Таблица 13

Значения разностей показателей эффективности для систем SMART и MEDLARS (усреднение по нескольким методам поиска, применяемым к системе SMART)

Вид словаря, используемого в SMART	Массив «расширенный MEDLARS»		Массив по офтальмологии					
			Внешняя экспертная оценка		Оценка релевантности дана авторами запросов (релевантными считаются документы, имеющие хоть какое-нибудь значение)		Оценка релевантности дана авторами запросов (релевантными считаются документы, имеющие важное значение)	
	П	Т	П	Т	П	Т	П	Т
Словарь «s»	−15,00%	−14,33%	−18,66%	−12,33%	+42,50%	−38,00%	+405,00%	−40,66%
Словарь основ	−16,00%	−12,66%	−12,00%	−12,33%	+23,50%	−35,66%	+337,66%	−39,66%
Автоматический список слов	—	—	−9,66%	+8,33%	+63,00%	−34,00%	+156,00%	−42,33%
Автоматический словарь слов-различителей	−2,33%	−3,00%	−2,00%	−13,66%	+108,25%	−52,00%	+247,00%	−52,33%
Тезаурус	+1,00%	−1,33%	−5,66%	−7,00%	+118,00%	−38,66%	+152,66%	−39,66%

документов, а также при поиске с обратной связью (см. табл. 12).

2) Итеративная стратегия поиска, реализованная в системе SMART, дает значительный выигрыш по сравнению с обычными методами поиска, используемыми в MEDLARS (см. табл. 12).

3) Лучшие показатели эффективности получаются, если использовать словарь слов-различителей и тезаурус, чем в том случае, когда применяются лишь процедуры распознавания основ или самих слов (см. табл. 13).

4) Методы нормализации языка документов, используемые в системе SMART для построения словарей и тезаурусов, в среднем дают по меньшей мере такую же эффективность поиска, что и обычное ручное индексирование документов (см. табл. 13).

5) Возможно, что в ИПС будущего будет использоваться метод сравнения векторов, который позволяет осуществлять ранжированную выдачу, а также итеративный поиск для получения улучшенных формулировок поисковых предписаний.

6) Возможно, что в будущем синтаксический и семантический анализ текстов, а также ввод документов, осуществляемый сейчас обычно опытными специалистами-индексаторами, будет заменен некоторыми средствами анализа, которые будут получены при помощи машины на основе уже имеющихся массивов документов.

ЛИТЕРАТУРА

1. Garvin P. L. et al., Some opinions concerning linguistics and information processing. Rep. PB 190 639, Center for Applied Linguistics, May 1969. Available from National Technical Information Service, Washington D. C.
2. Edmundson H. P., New methods in automatic extracting. *J. ACM* **16**, 2 (1969), 264—285.
3. Pacak M., and Pratt A. W., The function of semantics in automated language processing. Symp. on Information Storage and Retrieval, Univ. of Maryland, Apr. 1971.
4. Baxendale P., An empirical model for machine indexing. Third Institute on Information Storage and Retrieval, American U., Washington D. C., Feb. 1961, pp. 207—218.
5. Clarke D. C., and Wall R. E., An economical program for the limited parsing of English, Proc. AFIPS 1965 FJCC, Vol. 27, Pt. 1, Spartan Books, New York, pp. 307—319.
6. Damerau F. J., Automatic parsing for content analysis. *Comm. ACM* **13**, 6 (1970), 356—360.
7. Rush J. E., Salvador R. and Zamora A., Automatic abstracting and indexing: Production of indicative abstracts by application of contextual inference and syntactic coherence criteria. *J. ASIS* **22**, 4 (1971), 260—274.
8. Salton G., Automatic text analysis. Science 168, 3929 (17 Apr. 1970), 335—343.

9. Cleverdon C. W., and Keen E. M., Factors determing the performance of indexing systems; Vol. 2 — test results. Aslib Cranfield Res. Proj., Cranfield, England, 1966.
10. Salton G., and Lesk M. E., Computer evaluation of indexing and text processing. *J ACM*, **15, 1** (1968), 8—36.
11. Salton G., Automatic processing of foreign language documents. *J. ASIS*, **21, 3** (1970), 187—194.
12. Dennis S. F., The design and testing of a fully automatic indexing-searching system for documents consisting of expository text. In Information Retrieval — A Critical View, G. Schecter, Ed., Thompson Book Co., Washington D. S., 1967.
13. Giuliano V. E., and Jones P. E., Study and test of a methodology for laboratory evaluation of message retrieval systems. Rep. ESD-TR-66-405, Arthur D. Little, Cambridge, Mass., 1966.
14. Sparck Jones K., Automatic Keyword Classification for Information Retrieval. Butterworth and Co., London, 1971.
15. Stevens M. E., Automatic indexing: A state of the art report. NBS Monograph 91, U. S. Bureau of Standards, Washington D. C., March 1965.
16. Stevens M. E., Giuliano V. E., and Heilprin L. B. Statistical association methods for mechanized documentation. NBS Misc. Pub. 269, U. S. Bureau of Standards, Washington D. C., Dec. 1965.
17. Swanson D. R., Searching natural language text by computer. Science, 132, 3434 (Oct. 21, 1960), 1099—1104.
18. Swanson D. R., Interrogating a computer in natural language. Proc. IFIP Cong. 1962, North-Holland Publishing Co., Amsterdam, p. 288—393.
19. The Principles of Medlars. National Library of Medicine, Bethesda, Md., 1970. Available from Superintendent of Documents, Washington D. C.
20. Salton G., Automatic Information Organization and Retreival. MsGraw-Hill, New York, 1968. См. русский перевод: «Автоматическая обработка, хранение и поиск информации». Под ред. проф. А. И. Китова. М., «Сов. радио», 1973. 560 стр.
21. Salton G., The SMART Retrieval System — Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, N. J., 1971.
22. Salton G., A comparison between manual and automatic indexing methods. American Documentation 20, 1 (Jan. 1969), 67—71.
23. Salton G., A new comparison between conventional indexing (Medlars) and automatic text processing (SMART). *J. ASIS* **23, 2** (1972), 75—84.
24. Lancaster F. W., Evaluation of the Medlars demand search service. National Library of Medicine. Bethesda, Md., Jan. 1968.
25. Salton G., Search and retrieval experiments in real-time information retrieval. In Information Processing 68 (Proc. IFIP Cong.), North-Holland Publishing Company, Amsterdam, 1969, pp. 1082—1093.
26. Salton G., The performance of interactive information retrieval. Information Processing Letters 1, 2 (July 1971), 35—41.
27. Borko H., The construction of an empirically based mathematically derived classification system. Rep. SP-588, System Development Corp., Santa Monica, Calif., Oct. 1961.
28. Augustson J. G., and Minker J. An analysis of some graph theoretical cluster techniques. *J. ACM*, **17, 4** (1970), 571—588.
29. Doyle L. B. Breaking the cost barrier in automatic classification, Rep. SP-2516, System Development Corp., Santa Monica, Calif., July 1966.
30. Gotlieb C. C., and Kumar S., Semantic clustering of index terms. *J. ACM*, **15, 4** (1968), 493—513.
31. Dattola R. T., Experiments with a fast algorithm for automatic classification. In The SMART Retrieval System — Experiments in Automatic Document Processing, G. Salton, Ed., Prentice-Hall, Englewood Cliffs, N. J., 1971.

32. Johnson D. B., and Lafuente J. M., A controlled single-pass classification algorithm with application to multi-level clustering. Sci. Rep. ISR-18, Sec. XII, Dept. of Computer Science, Cornell U., Ithaca, N. Y., Oct 1970.
33. Bonwit K., and Aste Tonsman J., Negative dictionaries. Sci. Rep. ISR-18, Sec. VI, Dept of Computer Science, Cornell University. Ithaca, N Y., Oct. 1970.
34. Salton G., Experiments in automatic thesaurus construction for information retrieval Proc. IFIP Congress 71, Ljubljana, North-Holland Publishing Co., Amsterdam, 1972, pp. 115—123. См русский перевод: Эксперименты по автоматическому построению тезауруса для информационного поиска. Кибернетический сборник, № 11, М., «Мир», 1974.

Система аксиом для математической биологии[1])

Уильям Хоффман

Следующие шесть аксиом могут, по-видимому, обеспечить достаточную основу для построения универсальной и реалистической математической биологии: (1) Любой организм представляет собой (топологическое) объединение тканей. (2) Всякая ткань с математической точки зрения является конечным покрытием, окрестности которого суть клетки ткани. Всякая клетка ткани обладает определенной биологической активностью, математическим эквивалентом которой служит векторное (или тензорное) поле. (3) Ткань с топологической точки зрения можно рассматривать как компактное многообразие. (4) Псевдогруппа может служить математической моделью биологической формы и функции, причем (5) каждая из подобных псевдогрупп характеризуется инфинитезимальным порождающим оператором, представляющим математическую модель биологического действия клетки. (6) Организм приобретает биологические структуру и функцию, обеспечивающие оптимальность его функционирования с точки зрения трофических функций. Первая аксиома, в сущности, трюизм. Вторая легко устанавливается посредством «поименного» сопоставления с общепринятой биологической классификацией клеток. Вкупе аксиомы 1 и 2 утверждают эквивалентность развитой в биологии клеточной теории принципу, согласно которому клетка по отношению к биологическим форме и функции является инфинитезимальным порождающим оператором. Аксиома 3 вытекает из того, что число клеток в любой ткани необходимо конечно. Аксиомы 4 и 5 соответствуют констатации определяющей роли фибрилл и нитевидных фиброзных органелл, универсальности клеточного ядра, наличия «организационных центров» в морфогенезе и сохранения конкретных функций в условиях индивидуальной изменчивости. Кроме того, аксиома 5 отражает то обстоятельство, что анатомические структуры развиваются из более примитивных исходных посредством реализации небольших «дифференцирующих» изменений, а другими словами, посредством продол-

[1]) William C. Hoffman, A system of axioms for mathematical biology, Mathematical Biosciences, 16 (1973), 11—29.

жений соответствующих лиевых групп преобразований. Аксиома 6 практически перефразирует принцип адекватного проектирования Рашевского, прилагая его, однако, к его «естественной» области — трофическим функциям. Показано применение сформулированных принципов к процессу контактной индукции при эмбриональном развитии тканей и феномену аллометрического роста.

ВВЕДЕНИЕ

Процессы, с которыми сталкиваются биологические науки, представляют собой того или иного рода преобразования. Мышечное сокращение, кровоток, биохимические реакции, даже силы, вызываемые покровными и соединительными тканями, — все эти явления по природе своей являются векторными полями с локальными базисами. Как таковые они могут быть представлены математической моделью, записанной на языке групп преобразований:

$$G \times M \xrightarrow{\tau} M, \text{ определяемая операцией } \tau_g(x) = {}'x; \quad x, {}'x \in M,\ g \in G, \quad (1)$$

где G — непрерывная группа (обладающая стандартными групповыми свойствами замкнутости, ассоциативности, существования единицы и обратного элемента), M — многообразие (локально евклидово топологическое пространство), а преобразование τ_g характеризует определяющую операцию группы G на элементах $x, {}'x, \ldots$ многообразия M. Если множество элементов группы G не включает все обратные элементы для элементов из G, как это было бы, например, в случае необратимого процесса, то тогда группа G называется полугруппой преобразований.

Существует множество примеров подобных групп преобразований: фризы на фасадах зданий (группы переносов), снежинки и другие кристаллы (кристаллографические группы), двусторонняя симметрия тела животного (группа отражений) и другие симметрии, наблюдающиеся в растительном и животном мирах (диатомовые водоросли, простейшие, подсолнечники, филлотаксис, спиралевидные морские раковины и т. д.).

Группа представляет собой основной математический объект, обладающий как структурой (множество ее элементов), так и функцией (определяющая операция — закон композиции преобразований, образующих группу). В связи с этим группа исключительно удачно подходит для описания явлений, рассматриваемых в биохимии, физиологии, морфологии и психологии. В самом деле, единый математический принцип, который, по-видимому, обеспечивает общее представление феномена жизни, состоит

в том, что клетка, рассматриваемая в качестве основного компонента ткани, является так называемым ростком группы Ли. Более подробно этот принцип будет обсуждаться в следующем разделе.

Существо дела здесь состоит в эволюционном приспособлении биологических систем организмов к физическим и геометрическим характеристикам внешнего мира, в котором они себя «проявляют». Биомеханика движений скелета, «константности» психологии восприятия, биохимические универсалии жизненных процессов [8], движения и потоки, связанные с морфогенезом, — все это реакции отдельных видов организмов на соответствующие инвариантности, свойственные геометрико-физико-химическим характеристикам внешней среды, которые организмы «сумели» идентифицировать и включить в свою филогению в процессе эволюции. Чем больше инвариантных, регулярных свойств своего внешнего мира смог распознать и «учесть» организм, тем больше хаоса удается ему устранить из внешней среды, что в конце концов обеспечивает его преимущества с точки зрения принятия решений, уменьшения фрустрации, доминирования и, по существу, собственно выживания.

В данном контексте ключевым понятием является симметрия — слово со многими значениями; мы, однако, используем его в математическом смысле, т. е. как инвариантность относительно преобразования. Этот подход затрагивает такие периодические и упорядоченные свойства биологической среды, как форма, функция и биохимические процессы. В каждом конкретном случае, однако, соответствующее определяющее векторное (тензорное) поле можно задать, лишь хорошо зная динамику собственно физико-химического процесса. Выяснение последней обычно представляет собой совершенно отдельную задачу, предусматривающую анализ уравнений движения, уравнений химической кинетики и т. п., причем последние в свою очередь снова могут быть представлены в виде «собственных» векторных полей.

Согласно Холдену и Брегману [11]: «Распознавание менее явных (чем у классических групп) симметрий может часто выполнять аналогичную роль, причем иногда — в более широком смысле. В природе существует множество объектов, общие очертания которых являют симметрии, вызванные формообразующими профакторами. Обнаружение у объекта симметрии привлекает внимание к силам, вызвавшим ее появление. Частные отклонения от этой симметрии могут в таком случае быть изучены отдельно».

Подобные отклонения от основной симметрии (какую бы форму она ни принимала) часто имеют характер наложенных на нее случайных возмущений и могут рассматриваться как резуль-

тат естественного развития исходной нестатистической, детерминированной ситуации. При таких обстоятельствах данный подход дает возможность изучить основные проявления биологической формы и функции с тем пониманием, что последняя исчерпывается силами, движениями, потоками и биохимическими реакциями. Статистические флуктуации, неизбежно налагающиеся на основной процесс, следует рассматривать отдельно. Основным объектом в этой модели является так называемая структурно устойчивая динамическая система, свойства которой более подробно мы обсудим ниже.

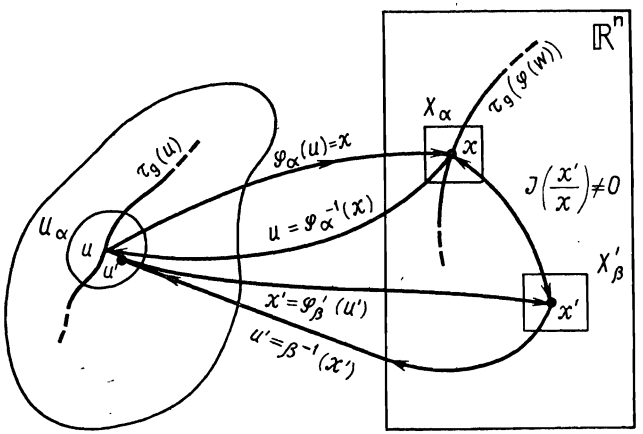


Рис. 1. Многообразие M гомеоморфно некоторому подмножеству n-мерного евклидова пространства $\mathbb{R}^n$. Отображение в евклидово пространство $\mathbb{R}^n$ таково, что якобиан $J(x'/x)\neq 0$. Траектория $\{\tau_g(u)\}$, проходящая через точку $u\in M$, соответствует преобразованию Ли $G\times M\xrightarrow{r} M$; это преобразование индуцирует соответствующую орбиту $\{\tau_g[\phi(u)]\}$ и в евклидовом пространстве $\mathbb{R}^n$.

Рассмотрим вкратце нашу математическую структуру, отложив детали до следующего раздела, в котором будут сформулированы основные постулаты. Группа Ли преобразований является одновременно дифференцируемым многообразием M (в двумерном случае обычной поверхностью) и группой, действие которой на многообразии должно быть «гладким» (т. е. непрерывной дифференцируемой) (рис. 1). Структура группы должна быть согласована со структурой дифференцируемого многообразия M.

Действие группы определяется локально векторным полем X и соответствующей производной Ли L_X [9, 15, 25]. В приложениях данной модели к биологическим феноменам многообразие M, так же как и группа G, является компактным. Свойство

компактности означает, что существует конечное покрытие многообразия окрестностями. В нашем случае это свойство представляется тем фактом, что ткань состоит из клеток. Другим примером могли бы послужить подобласти, образующие многообразие зрительного восприятия и имеющие диаметр, по порядку соответствующий пределу остроты зрения [10]. Каждая из таких окрестностей является «локально евклидовой» в том смысле, что существует взаимное дифференцируемое отображение на соответствующую подобласть евклидова пространства (на рис. 1 эти отображения указаны стрелками). С точки зрения биологических приложений эта особенность математической структуры означает, что весь теоретический анализ вплоть до «минуты откровения» может проводиться исключительно в пределах более удобного евклидова пространства $\mathbb{R}^n$, и только на этом этапе, но отнюдь не раньше, следует сопоставлять теоретические выводы (предсказания) с реальными характеристиками исследуемого явления.

Локальное действие группы Ли заключается в том, чтобы соединить соседние точки многообразия лежащей на нем кривой, называемой *орбитой*, *траекторией* или *линией тока* (обтекания); форма этой кривой определяется свойствами конкретной группы преобразований. Согласно первой основной теореме Ли, группа Ли преобразований в евклидовом пространстве $\mathbb{R}^n$ полностью определяется так называемой производной Ли

$$L_X = \sum_{i=1}^{n} X_i(x, \ldots, x) \frac{\partial}{\partial x_i}, \tag{2}$$

представляющей собой, по крайней мере в случае обычных функций, обыкновенную производную по направлению, берущуюся в направлении касательной к орбите группы во всех точках последней. В связи с этим и говорят, что действие лиевой производной заключается в «протаскивании потока» [9] по орбите соответствующей группы Ли.

Математический «поток» определяется динамической системой. Последняя является однопараметрической группой, представляющей интегральное многообразие (или, в случае двумерного евклидова пространства $\mathbb{R}^2$, кривую) системы обыкновенных дифференциальных уравнений первого порядка [6, 9, 25]. Такая система дифференциальных уравнений называется системой Пфаффа. Согласно известной теореме из теории дифференциальных уравнений [3], решениями такой системы являются частные решения дифференциального уравнения Лагранжа в частных производных, представляющие инвариантность определяющей операции группы Ли

$$L_X F(x_1, \ldots, x_n) = 0.$$

Следовательно, применение операции взятия производной Ли к любой функции, инвариантной относительно определяющей операции группы Ли, должно приводить к получению нуля.

В то же время алгебра Ли (скобочные произведения), соответствующая группе Ли, индуцирует присоединенное векторное поле X. В таком случае двойственность обеспечивает существование соответствующего множества дифференциальных форм, также определяющих интегральные кривые (орбиты). С другой стороны, те биологические функции, представление которых в связи с их «векторно-полевой» природой оказывается более естественным на языке локальных объектов, можно описывать такими моделями, если речь идет о тканях или иных элементах укрупненной биологической структуры. Так, например, биохимические воздействия, локальные потоки, механические силы, порожденные биологическими причинами, — все эти феномены более естественно отнести к общему классу векторных полей.

АКСИОМАТИЧЕСКОЕ ВВЕДЕНИЕ ФУНДАМЕНТАЛЬНЫХ БИОЛОГИЧЕСКИХ ПРИНЦИПОВ

Мы выдвигаем несколько математических аксиом, которые по существу являются переформулировкой хорошо известных биологических принципов. Опираясь на них, мы получим с помощью как математических, так и биологических доводов ряд фундаментальных биоматематических результатов. Последние порождают математическую структуру, которая, судя по всему, охватывает общую теорию биологической формы и функции.

Несколько злоупотребляя терминологией, мы будем непринужденно переходить от «группы (преобразований) Ли» к «псевдогруппе» и «псевдополугруппе», в большей или меньшей степени руководствуясь контекстом. Если исследуемое явление имеет необратимый характер или характеризуется бесконечным множеством существенных параметров, то в этих случаях действительно появляется либо псевдогруппа, либо псевдополугруппа.

Аксиома 1. *Любой орган (организм) представляет собой объединение тканей*

$$\Omega_i = \sum_{j=1}^{n} M_{ij}, \tag{3}$$

где M_{ij} — хаусдорфово пространство (не обязательно связное).

Аксиома 2. *Каждая ткань M_j представляется конечным покрытием клетками определенного типа (или типов), рассматриваемыми в сочетании с их соответственной деятельностью. Эта интегральная деятельность порождает некоторое преобразование,*

характеризующееся соответствующим векторным (или тензорным) полем

$$M_j = \sum_{k=1}^{n} U_{jk}; \quad T : U_{jk} \to V_{jk}. \tag{4}$$

Первая часть аксиомы 2 по существу перефразирует основное положение развитой в биологии клеточной теории, сводящееся к тому, что клетка является основой жизни [23]. Таким образом, подмножество окрестностей пространства (ткани) M_j реализуется на клетках ткани. Вышеупомянутую деятельность клетки обозначим через T — стандартное обозначение функтора по направлению. Введение последнего определяется тем обстоятельством, что биологическая функция, осуществляемая клеткой U_{jk} (или появляющаяся в результате ее деятельности), порождает векторное поле V_{jk}; оно представляет силы, развиваемые клеткой (если они имеются), ее возможные движения, потоки, поступающие из клетки наружу, биохимические реакции, происходящие в клетке, и т. д.

Аксиома 3. *Ткань с топологической точки зрения можно рассматривать как компактное многообразие.*

В таком случае принимается, что упоминавшееся в аксиоме 2 покрытие является конечным (для любой ткани это в высшей степени разумное допущение), а хаусдорфово пространство аксиомы 1 предполагается локально евклидовым. Другими словами, для любой входящей в покрытие окрестности U_k n-мерного многообразия M существует гомеоморфизм ϕ_k, отображающий окрестность U_k на открытое множество E_k евклидова пространства R^n:

$$\phi_k : U_k \to E_k \subset \mathbb{R}^n. \tag{5}$$

Кроме того, функции $\phi_k \phi_l^{-1}$ и $\phi_l \phi_k^{-1}$ определяют на пересечении $U_k \cap U_l$ двух подмножеств многообразия M гомеоморфизмы открытых множеств $\phi_k(U_k \cap U_l)$ и $\phi_l(U_k \cap U_l)$ в евклидово пространство R^n. На языке клеточных механизмов данное свойство обеспечивает взаимную согласованность тонких клеточных процессов и возможность переключения деятельности клетки с одного процесса на другой.

Следуя Тондьюру [27], не будем требовать обязательной связности многообразия.

Свойства введенного выше гомеоморфизма (5) естественно приводят к понятиям псевдогруппы, карты и атласа, которые будут нужны нам в процессе дальнейшего обсуждения.

Определение псевдогруппы [15, 21]: *псевдогруппа преобразований* на топологическом пространстве S представляет собой

совокупность Γ гомеоморфизмов f открытых подмножеств топологического пространства S (*областей определения* гомеоморфизмов f) в открытые множества топологического пространства S (*множества значений* f), такую, что

(1) совокупность Γ замкнута относительно сужения:

$$f\colon U_1 \to S;\quad U_2 \subset U_1;\quad U_2 \text{ — открытое} \Rightarrow f|_{U_2} \in \Gamma;$$

(2) элементы совокупности гомеоморфизмов Γ можно „склеивать" в единое целое: если $U = \bigcup\limits_{\alpha} U_\alpha$ и

$f\colon U \to S$, то $f \in \Gamma$ при условии, что $f|_{U_\alpha} \in \Gamma$ для всех α;

(3) совокупность гомеоморфизмов Γ замкнута относительно *композиции*: $f_1, f_2 \in \Gamma$, $f_1\colon U_1 \to U_2$;

$$U_3 \to U_4;\quad U_2 \cap U_3 \neq \varnothing \Rightarrow f_2 \circ f_1 \in \Gamma,$$

где $f_2 \circ f_1\colon f_1^{-1}(U_2 \cap U_3) \to f_2^{-1}(U_2 \cap U_3)$;

(4) *тождественный* гомеоморфизм входит в совокупность гомеоморфизмов Γ: для любого открытого множества

$$U \subset S,\quad \exists \mathrm{id}\colon U \to U,\quad \mathrm{id} \in \Gamma.$$

(5) совокупность гомеоморфизмов Γ замкнута относительно *обратных* отображений гомеоморфизмов

$$f \in \Gamma : f \in \Gamma \Rightarrow f^{-1} \in \Gamma.$$

Если условие (5) не выполняется, то это означает, что мы имеем дело с псевдополугруппой, а не с псевдогруппой.

Если гомеоморфизмы f представляют собой диффеоморфизмы и топологическое пространство S является носителем дифференцируемой структуры, то понятие псевдогруппы есть обобщение группы преобразований Ли. Следовательно, дифференцируемая псевдогруппа образуется совокупностью диффеоморфизмов, локально определенных на многообразии; совокупность элементов группы замкнута относительно операций композиции и получения обратного элемента, если эти преобразования имеют содержательный эквивалент — т. е. область определения второго диффеоморфизма не перекрывает множества значений предшествующего диффеоморфизма.

Пара (U_k, φ_k), состоящая из окрестности, входящей в покрытие многообразия M, и соответствующего диффеоморфизма в евклидово пространство $\mathbb{R}^n$, образует так называемую *карту* (или «локальную карту», «согласованную окрестность») на многообразии M. *Атлас* дифференцируемого многообразия M, совместный с некоторой псевдогруппой Γ, представляет собой множество карт, таких, что соответствующие окрестности U_i

образуют покрытие многообразия M. В таком случае из аксиомы 3 непосредственно вытекает следующее положение:

Принцип 3 — 1. *Карта представляет клетку, локализованную в каком-либо конкретном участке ткани. Ансамбль образующих ткань клеток представляется атласом соответствующего многообразия.*

Установив содержательные эквиваленты понятий карты и атласа применительно к нашей проблеме, можно перейти к формулировке аксиомы 4:

Аксиома 4. *Псевдогруппа может служить математической моделью биологической формы и функции.*

Другими словами, во введенных выше обобщенных представлениях тканей как многообразий и биологической активности как преобразований над тканями, определяемых векторными полями, соответствующие многообразия и действия таковы, что задают псевдогруппу, адекватную исследуемому биологическому феномену. Локальные преобразования, введенные аксиомой 2, не только обеспечивают воздействие на образующие карту окрестности, но также предусматривают существование ассоциированной параметрической группы, действие которой заключается в установлении объема и темпа преобразования [7, 18]. Следующее замечание Боннера прекрасно характеризует действие параметрической группы [4, стр. 271]: «...паттерн, к установлению которого привело новое развитие, можно рассматривать как результат взаимодействия образующих процессов с процессами, ограничивающими развитие».

Последние, т. е. эти «процессы, ограничивающие развитие», образуют параметрические группы, ассоциированные с орбитальными потоками («образующими процессами») данной группы преобразований. По этому поводу мы рекомендуем также обратиться к монографии Уоддингтона [28, стр. 104].

Итак, мы можем идентифицировать биологические воздействия и движения, в том числе биохимические реакции, поставив им в соответствие действия некоторых групп (псевдогрупп) преобразований. Отсюда следует, что биологические потоки можно отождествить с «математическими потоками» (однопараметрические группы преобразований). *Локальная* структура группы преобразований, определяемая соответствующими векторными полями, производными Ли и дифференциальными формами, отождествляется с гистологией или, проще говоря, микроскопическими элементами, образующими соответствующую ткань. Аналогичным образом общий характер псевдогруппы как независимого целостного объекта отождествляется с поведением соб-

ственно ткани как макрообъекта в сочетании с ее взаимодействием с другими тканями.

Принцип 4—1. *Биологическая форма определяется движением клеток по орбитам соответствующих псевдогрупп.*

Обсуждение. Орбита представляет собой траекторию (линию тока), прокладываемую по многообразию посредством реализации преобразований, индуцированных действием псевдогруппы [27, определение 1.3.1]. Следовательно, отдельные связные компоненты образуют связанные траекторией подмногообразия многообразия M, которые представляют разбиение последнего [27, лемма 1.3.2]. Итак, математическое содержание принципа 4—1 непосредственно следует из аксиом 3 и 4 и упомянутой леммы.

Что касается биологического обоснования принципа 4—1, мы рекомендуем читателю обратиться к: (i) соответствующим местам монографии Уоддингтона [28, стр. 81], обсуждающим движение ткани в бластодерме птенца, движение клетки в эмбрионе морского ежа (там же, стр. 97), истечения при гаструляции в яйцеклетках амфибии (там же, стр. 99); (ii) главам IV, V и IX монографии Томпсона [26], в которых приведено много примеров биологической формы в переводе на язык групповых орбит; (iii) статье Стала [24], посвященной изучению процессов новообразования с точки зрения преобразований подобия. Отличные статьи фон Энглхардта, Тролла и Вулфа в журнале «Studium Generale» (июль 1949 года) по поводу роли симметрии в биологической форме также служат поддержкой нашего принципа. Можно было бы привести множество других примеров биологических форм, соответствующих орбитам разного рода потоков и псевдогрупп, однако вышеупомянутых ссылок должно быть достаточно для установления реалистичности нашего принципа с биологической точки зрения.

Принцип 4 — 1.1. *Биологическая форма с точностью до статистических флуктуаций («индивидуальная изменчивость») определяется инвариантностью относительно действия соответствующей псевдогруппы.*

Обсуждение. Этот принцип представляет собой то, что на языке математики называется непосредственным следствием, в данном случае — принципа 4—1 и лемм 1.3.2 и 1.3.3 из монографии Тондьюра [27]. По поводу биологического смысла настоящего принципа мы рекомендуем читателю обратиться к соответствующим местам монографии Беррилла [2, стр. 17 и 109 и следующие за ними], касающимся неизменности конфигураций волокон ресничек и жгутиков, встречающихся в биологической структуре простейших, и подавления клеточной индивидуальности однородностью тканей; к описанию дифференциации тканей

in vitro в монографии Уоддингтона [28, стр. 51] (а также к работе Уиллмера [29], посвященной трем основным морфологическим типам клеток, выращенных in vitro); к экспериментальному исследованию направленных микроскопических канальцев протяженных клеток, предпринятому Байерсом и Портером [5]; к исследованию формирования мышечного тракта в миогенезе, выполненному Конигсбергом [16]; к описанию процесса инвагинации, данному Берриллом [2, стр. 140] с позиций клеточных механизмов; к его же [2, стр. 15] анализу первичности роли фибрилл и экстрацеллюлярных (внеклеточных) систем волокон [2, стр. 22]; к тем основным схемам развития многоклеточных организмов: радиальной, с преобладанием двусторонней симметрии и с многократным повторением одной и той же структуры, как это имеет место, например, у сегментированных животных, организмов-колоний и травянистых растений. Классификация Хоумса [12, глава II] типов клеточной специализации также представляет собой сильный аргумент в пользу нашего принципа. Шмитт указал [20], что, по всей вероятности, именно две разновидности фиброзных органелл, нитевидные и капиллярные, являются во всей живой природе основными структурными элементами клетки. Кроме того, мы хотели обратить внимание на точку зрения Беррилла [2, стр. 16], согласно которой фиброзные белки, весьма вероятно, являются структурной основой организации клеток и тканей.

Принцип 4—1.2. *Отдельные элементы ткани воспроизводят неподвижные точки функции, представляющей действие ткани, или, другими словами, соответствуют группе (подгруппе) изотропии псевдогруппы.*

Обсуждение. Согласно аксиомам 2 и 4, можно считать, что ткань представляется дифференцируемым многообразием и рядом соответствующих векторных полей. Опираясь на известную теорему [1, стр. 550] о существовании конечного множества особых точек векторного поля, определенных на подобном дифференцируемом многообразии, можно прийти к выводу о существовании лишь конечного числа точек, характеризующих тканевую функцию. Это же заключение следует из того обстоятельства, что множество неподвижных точек составляет подмногообразие (не обязательно связное) группы преобразований [17], а также и из представления псевдогруппы как G-структуры [22, стр. 120]; [27, стр. 1].

Для полноты введем определение группы (подгруппы) изотропии: *группа (подгруппа) изотропии* группы преобразований $G \times M \to M$ для элементов многообразия $u_0 \in M$ состоит из таких элементов группы преобразований $g \in G$, при которых дей-

ствие группы

$$G_{u_0} = \{g \in G:\ \tau_g(u_0) = u_0,\ u_0 \in M\} \tag{6}$$

не изменяет статуса элементов многообразия u_0 как неподвижных точек. *Группа* (*подгруппа*) *изотропии* полной группы преобразований представляет собой прямое произведение групп (подгрупп) изотропии, определенное на всех точках многообразия M. Преобразование (группу преобразований) называют *свободным*, если G_{u_0} сводится к тождественному преобразованию на G, и обозначают 1_G. Нас, однако, интересуют группы изотропии более общего характера, а именно — соответствующие особым точкам векторного поля, принадлежащим многообразию M. В таком случае вторая часть принципа 4—1.2 становится очевидной. Первая (математическая) часть принципа непосредственно следует из того обстоятельства, что неподвижные точки групповой операции τ

$$\tau_g(u) = {}'u, \quad u \in U \subset M, \quad {}'u \in {}'U \subset M \tag{7}$$

определяются преобразованием

$$\tau_g(u) = u, \tag{8}$$

следовательно, при выполнении условия (8) не только преобразование g является элементом подгруппы изотропии, но и элемент многообразия u оказывается точкой покоя (неподвижной точкой) группы преобразований.

Биологическое содержание принципа 4—1.2 обосновывается либо тем обстоятельством, что клетка представляет собой некоторый элементарный орган (в плане локализации или функции), характеризующийся разнообразными внутриклеточными процессами и обладающий способностью перемещаться, образовывать новые клетки, порождать силы и т. д. (см. монографию Беррилла, [2, стр. 17, 21—23, 58, 59, 139]), т. е. клетка является неподвижной точкой относительно своей функции, либо, с другой стороны, тем фактом, что клетка сама содержит отдельные элементы, скажем такие, как ядра или ядрышки, которые можно рассматривать в качестве неподвижных точек биологической функции клетки (см. монографии Уоддингтона [29, стр. 18] и Беррилла [2, стр. 20]). В качестве других биологических объектов, которые можно отождествить с неподвижными точками (точками покоя соответствующих преобразований), отметим уоддингтоновские «эвокаторы» [28, стр. 77] и спеманновские (Spemann) организационные центры, полюса и т. п. «образования», проявляющие свое действие в процессе морфогенеза.

Принцип 4—1.3. *Биологическая структура и функция представляют собой реализацию пучка классов смежности.*

Обсуждение. Согласно аксиоме 4, биологическая структура и функция представляют собой реализацию групп

преобразований. Последние всегда поддаются факторизации по пространствам траекторий, причем каноническое отображение имеет вид [27, стр. 14, лемма 1.3.2]

$$\pi_\Omega\colon\ \Omega \to \Omega/G, \tag{9}$$

где Ω/G — «траекторное» множество Ω, характеризующее его разбиение на непересекающиеся подмножества. Итак, налицо все характерные элементы пучка класса смежности: расслоенное пространство $\Omega \times G$, проекция π_Ω, базисное пространство Ω/G и групповая структура G. Как указывалось выше, первый из этих элементов обеспечивает воспроизведение формы и функции, а проекция и соответствующая факторизация — специализацию тканей и клеточной морфологии, определяемую характером рассматриваемого биологического явления.

Принцип 4—1.4. *Биологическая форма представляется инвариантностью относительно действия почти свободной структурно устойчивой группы преобразований.*

Обсуждение. Свободная группа преобразований относится к числу тех групповых объектов, группа изотропии которых является тривиальной, т. е. состоит только из единичного элемента и всей группы. Действие группы называется *почти свободным*, если группа изотропии строго дискретна; следовательно, группа преобразований является почти свободной, если ее неподвижные точки разделены. В таком случае множество неподвижных точек компактного многообразия является дискретным и конечным. В данном контексте, после того как мы отождествили неподвижные точки с биологическим понятием центра, становится очевидной биологическая значимость условия «быть почти свободной».

Смысл понятия «структурная устойчивость» сводится к тому, что топологические свойства фазового пространства остаются неизменными при воздействии малых возмущений на динамическую систему, являющуюся «источником» фазового портрета. Используя известные соответствия между динамической системой и системой уравнений Пфаффа, а затем — между последней и дифференциальным уравнением Лагранжа в частных производных, приходим к соответствию вида

$$\frac{d\mathbf{x}}{dt} = \mathbf{X}(\mathbf{x}) \leftrightarrow L_{\mathbf{X}} = \mathbf{X} \cdot \nabla \tag{10}$$

или в наиболее общем виде:

$$\begin{array}{ccc} & M & \\ {\scriptstyle\pi}\swarrow & & \searrow{\scriptstyle\pi^*} \\ TM & \xleftrightarrow[\phi^*]{} & T^*M \end{array}$$

Оба построения, таким образом, при данных начальных условиях приводят к одной и той же конфигурации траекторий. Если система $\mathbf{X}(x)$ подвергается воздействию малого возмущения, под влиянием которого она переходит в некоторое состояние $\mathbf{X}(x)+\mathbf{P}(x)$, предположение о наличии структурной устойчивости обеспечивает сохранение общего характера топологии пространства решений уравнения (10) в том смысле, что в подпространствах, входящих в фазовый портрет системы, будут наблюдаться траектории тех же типов. Отсюда вытекает непосредственная связь с явлением индивидуальной изменчивости. Концепция индивидуальной изменчивости сводится к тому, что отдельные особи, принадлежащие одному и тому же или родственным видам, обладают одной и той же биологической функцией, несмотря на значительные структурные различия системы органов, обеспечивающих выполнение соответствующей функции. «Головной мозг разных людей столь же неодинаков, как и их лица, их желудки, их печени», тем не менее основные механизмы восприятия и познания у всех нас одинаковы, если только они не нарушены каким-либо поражением, приводящим к негомеоморфному изменению структуры.

Аксиома 5. *Любая группа (псевдогруппа) преобразований, введенная в аксиоме* 4, *имеет инфинитезимальный порождающий оператор, представляющий математическую модель биологического действия клетки.*

Аксиома 5 вводит обобщенное представление, по меньшей мере локальное, биологического механизма в виде действия инфинитезимального порождающего оператора на дифференцируемом многообразии; этот инфинитезимальный порождающий оператор представляет собой лиеву производную $L_{V_{jk}}$, определяемую соответствующим векторным полем (аксиома 2, соотношение (4)). Математическое обоснование выбора такой модели можно найти в монографии Стернберга [25, стр. 89—90] и статье Сингера и Стернберга [21, стр. 10]. Интересующимся биологическим обоснованием модели рекомендуем обратиться к монографии Уоддингтона [28, стр. 9], в частности таким положениям, как:

«морфогенез ... требует объяснения в терминах ... сил, переводящих ... вещество из исходной формы в новую»;

концепция «эмбрионального поля» (стр. 16 и 96):

«... некоторые клеточные взаимодействия осуществляются через специализированные зоны на поверхности клетки, которые оформляются в так называемые «тела прикрепления» или десмосомы».

Что касается развившейся клетки, то ее активность мы рассматриваем в категориях развиваемых ею сил, совершаемых ею перемещений и реализуемых биохимических реакций. Все эти процессы происходят на микроуровне, и каждому из них можно поставить в соответствие некоторое векторное поле. В целом это явление обладает всеми характерными свойствами инфинитезимального порождающего оператора.

Принцип 5—1. *Клетка, рассматриваемая в качестве основного компонента ткани, представляется ростком лиевой группы преобразований.*

Обсуждение. Для заданной окрестности единичного элемента лиевой группы преобразований соответствующий росток группы Ли определяется как класс эквивалентности над всеми группами Ли преобразований, имеющими одно и то же инфинитезимальное преобразование (а следовательно, и одну и ту же производную Ли) в этой окрестности. Согласно первой основной теореме Ли, росток группы Ли полностью определяется соответствующей лиевой производной. Из аксиом 4 и 5, однако, следует, что производная Ли представляет собой инфинитезимальный порождающий оператор, который может служить моделью биологических процессов, связанных с деятельностью на микроуровне клеток различных типов. В таком случае предложенная нами математическая модель клетки задает вполне определенный класс локальной эквивалентности, соответствующий заданной окрестности и производной Ли. Следовательно, росток группы Ли определен, если выполнены внутренние условия совместности для ростка группы G_e и алгебры Ли LG. Математическое обоснование обеспечивается второй основной теоремой Ли и соответствующей обратной теоремой, т. е. тем фактом, что некоторая система линейно независимых производных Ли порождает (замкнутую) алгебру Ли в том и только том случае, если скобочные произведения удовлетворяют тождественному преобразованию

$$[Lx_i,\ Lx_j]=\sum_{k=1}^{r} c_{ij}^{k} Lx_k, \qquad (i,\ j=1,\ \ldots,\ r), \tag{11}$$

т. е. произведения Ли «привязаны» к линейному многообразию, покрытому производными Ли. На биологическом языке это означает, что клеточные процессы и механизмы на микроуровне «почти линейны», т. е. любой процесс может реализовываться в произвольной последовательности с любыми двумя другими процессами, причем результат все равно будет принадлежать тому же самому (локальному) линейному многообразию (положение или биологическое воздействие), не говоря уже об инфинитезимальных объектах высшего порядка. Судя по всему,

клетки, из которых образованы ткани, относятся к этой же категории, потому форма или действие клетки должны быть «интегрируемы» в том смысле, чтобы допускалась экстраполяция характерных свойств отдельной клетки на глобальную совокупность клеток, образующую ткань. В случае таких клеток, как кровяные, клетки защитительного действия и другие, не относящиеся к тканеобразующим, можно воспользоваться тем обстоятельством, что их активность реализуется на микроуровне и, следовательно, говоря математически, либо может быть линеаризована, либо является специфической и независимой. В любом случае они «заполнят» локальное линейное многообразие типа определенного правой частью уравнения (11). Что касается морфологии подобных клеток, то их очертания отличаются «гладкостью» (по меньшей мере, дважды дифференцируемы) и, следовательно, могут быть представлены поверхностью вида $\phi(x, y, z) = \text{const}$. Последняя есть решение дифференциального уравнения в частных производных для производной Ли

$$L_X\phi = 0$$

и, следовательно, ϕ — функция, представляющая траектории клеточной активности.

Принцип 5—1.1. *Биологическая структура представляет собой реализацию лиевой алгебры группы (псевдогруппы) преобразований.*

Обсуждение. Первая основная теорема Ли утверждает, что лиева группа преобразований полностью определяется своими производными Ли. Последние, как известно, образуют алгебру Ли; кроме того, они порождают траектории группы преобразований, а в соответствии с принципом 4—1 биологическая форма определяется траекторным движением клеток.

Принцип 5—2. *Морфогенез и постнатальное развитие происходят в результате уточнения существующей биологической формы и функции; эти уточнения соответствуют введению дифференциальных инвариантов посредством продолжения производных Ли группы преобразований.*

Обсуждение. Ранние стадии эмбрионального развития отличаются наличием нескольких основных видов симметрии, которые при любых обстоятельствах являются реализацией отдельных траекторий полной линейной группы. Вскоре, однако, эти примитивные типы линейной, осевой и спиральной симметрии вытесняются более сложными формами, что обеспечивается реализацией непрерывной последовательности малых дифференцирующих переходов, уводящих от исходной структуры. Исходя из того, что производные Ли порождают первоначальную

базовую структуру траекторий (аксиома 5), уточнение ее должно определяться продолжением соответствующих производных Ли. Проиллюстрируем эти положения на конкретном примере: контактная индукция в тканях.

Понятие контактной индукции в тканях характеризует такое взаимодействие между различными частями ткани эмбриона, при котором развитие в одной части (*индукторе*) приводит к изменению характера развития и дифференциации в другой части [14]. Так, например, входящие в состав центральной нервной системы ткани, образующиеся из гаструлярной эктодермы, представляют собой первую группу эктодермальных органов, испытывающих такого рода влияние, причем в роли ткани-индуктора, являющегося источником этого влияния, выступает хордо-мезодерма («центр-организатор» по Спеманну). Центральная нервная система в свою очередь превращается в индуктор по отношению к развитию других эктодермальных органов, например, таких, как носовой мешок, хрусталик глаза и внутренняя структура уха.

Реализация морфогенеза и клеточная дифференциация приводят к изменению взаимного расположения тканей в процессе развития, что вовлекает в процесс индукции различные системы орган — индуктор. Реакция ткани зависит от: (I) «мощности» индуктора; (II) общего итога всех предшествующих индукционных воздействий; (III) продолжительности периода времени, в течение которого ткань находится в контакте с индуктором (индукторами), и (IV) числа индукторов. Если несколько индукторов действуют одновременно, эффекты, производимые ими, суммируются. Любая система индуктор — орган характеризуется градиентами индукционной способности (компетенцией), определяющими предрасположение данной ткани к формированию соответствующего органа. Часто о наличии индукции можно судить по изменениям формы и способа объединения клеток, как, например, в плакодах, «стоящих у истоков» развития носа, хрусталика и уха, или по усложнению видов движения и истечения клеток, как это происходит, в частности, при формировании мозговой пластинки.

Итак, ряд особенностей процесса индукции в тканях эмбриона весьма точно соответствует характерным свойствам (локальной) однопараметрической группы преобразований, индуцированной векторным полем, т. е. группы Ли преобразований. Продолжительность процесса индукции определяется соответствующими параметрическими группами. Индукторы, действие которых проявляется в виде «градиентных полей», являются локально аддитивными, т. е. обладают свойством, вполне недвусмысленно свидетельствующем о действии линейной комбинации лиевых производных.

Нижеследующая простая модель (плоское сечение) воспроизводит основные особенности явления индукции в тканях. Рассмотрим совокупность горизонтально движущихся клеток; передние образуют (локально) «плоскую» ткань. Это движение можно представить с помощью производной Ли:

$$L_1 = -y\frac{\partial}{\partial x}. \tag{12}$$

Предположим, что еще одна ткань такого же рода движется вертикально; этот поток (движение ткани) определяется следующей лиевой производной:

$$L_2 = x\frac{\partial}{\partial y}. \tag{13}$$

Траектории производной L_1 представляют собой горизонтальные прямые, траектории производной L_2 — вертикальные (рис. 2).

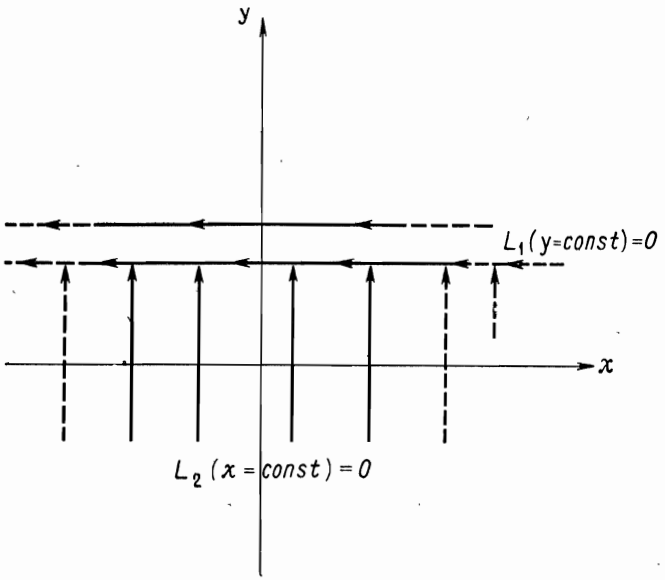


Рис. 2. Траектории движения, представляемые производными Ли $L_1 = -y(\partial/\partial x)$ и $L_2 = x(\partial/\partial y)$ (соответственно $y = \text{const}$ и $x = \text{const}$), до возникновения контактной индукции, определяемой результирующей производной Ли $L_O = -y(\partial/\partial x) + x(\partial/\partial y)$, т. е. движения по окружности.

Допустим, что наши ткани вступили в соприкосновение и вторая выполняет по отношению к первой роль индуктора. Можно ожидать, что в «зоне контакта» оба движения (и соответствующие производные Ли) будут аддитивно объединяться, по крайней

мере локально, что приведет к появлению результирующей лиевой производной:

$$L_{O_2} = -y\frac{\partial}{\partial x} + x\frac{\partial}{\partial y}. \tag{14}$$

Данная лиева производная характеризует действие группы вращений плоскости. Траектории, соответствующие этой лиевой производной, представляют собой семейство концентрических окружностей. Если для объединения лиевых производных, представляющих исходные движения, использована *взвешенная* комбинация вида

$$\varkappa_1 L_1 + \varkappa_2 L_2 = -\varkappa_1 y\frac{\partial}{\partial x} + \varkappa_2 x\frac{\partial}{\partial y}, \quad \varkappa_1,\ \varkappa_2 > 0, \tag{15}$$

то в результате должно появиться семейство эллиптических траекторий, скажем, типа тех, что наблюдаются в плакодах. Параметры $\varkappa_1$ и $\varkappa_2$, входящие в уравнение (15), пропорциональны значениям групповых параметров, определяющих интенсивность и продолжительность процесса индукции [6, приложение, теорема 1].

Итак, взаимодействие «плоской» ткани-индуктора и «плоской» ткани-индукта должно привести к образованию «скривленной» ткани. Степень возникших в результате локальных закручиваний и изгибов определяется продолжительностью взаимодействия тканей. Следовательно, изменения положения (по переменным x и y) в первом приближении пропорциональны продолжительности контакта $\delta\tau$:

$$\delta(x,\ y) = L_{O_2}(x,\ y)\cdot\delta\tau + o(\delta\tau).$$

В то же время при более продолжительном контакте r преобразование ткани определяется экспоненциальным отображением:

$$('x,\ 'y) = \exp(\tau L_{O_2})(x,\ y).$$

Последовательные конфигурации (тонкие пластинки) плакоды соответствуют отдельным элементам семейства кривых, характеризущимся радиальной симметрией. В таком случае произвольная функция F, определяемая параметрами $x^2 + y^2$ и $(y - xy')/(x + yy')$, т. е.

$$F\left(x^2 + y^2,\ \frac{y - xy'}{x + yy'}\right) = 0 \tag{16}$$

(где $x^2 + y^2$ — квадрат радиуса-вектора, а $(y - xy')/(x + yy')$, $[y' = (dy/dx)]$, — проекция радиуса-вектора на нормаль к кривой $y(x)$ в точке x), инвариантна относительно действия производной Ли, подвергнутой однократному продолжению:

$$L_{O_2}^{(1)} = L_{O_2} + (1 + y'^2)\frac{\partial}{\partial y'}. \tag{17}$$

Другими словами, любая форма, определяемая только двумя переменными — расстоянием до некоторого полюса («организационного центра») и расстоянием, взятом по нормали или касательной от того же самого полюса [1]), инвариантна относительно действия группы вращений плоскости O_2, подвергнутой однократному продолжению. Любое семейство тканей, обладающее радиальной симметрией, относится, как отмечалось выше, к этой категории.

Инфинитезимальные порождающие операторы $x(\partial/\partial x)$ и $y(\partial/\partial y)$ также воспроизводят прямолинейные движения, как в случае продвижения «прямолинейных» тканей. Совместное действие этих производных Ли будет, как и при возникновении контактной индукции, приводить к появлению сжатий и растяжений, т. е. группы растяжений плоскости. Растяжения эти оказываются неравномерными, если объединение представляется взвешенной комбинацией

$$L_d = \alpha x \frac{\partial}{\partial x} + \beta y \frac{\partial}{\partial y}. \tag{18}$$

Оказывается, что большинство клеток в тканях, участвующих в контактной индукции, образуют поток в направлении, поперечном относительно локальной ориентации самих тканей. Уравнение (18) описывает любые движения такого рода, происходящие в плакоде или инвагинации. Кроме того, уравнения (14) и (18), взятые вместе, определяют спиральные движения, также часто возникающие при росте ткани.

Первое продолжение производной L_d, заданной уравнением (18), определяется следующим выражением [6, таблица II]:

$$L_d^{(1)} = L_d + (\beta - \alpha)\, y' \frac{\partial}{\partial y'} \tag{19}$$

и в общем случае при $n = 1, 2, \ldots$ выражением вида

$$L_d^{(n)} = L_d + \sum_{k=1}^{n} (\beta - k\alpha)\, y^{(k)} \frac{\partial}{\partial y^k}. \tag{19а}$$

В наиболее общем виде дифференциальное уравнение, инвариантное относительно действия n раз продолженной лиевой производной $L_d^{(n)}$, определяется выражением $L_d^{(n)} F(x, y, y', \ldots, y^{(n)}) = 0$, что эквивалентно задаче нахождения произвольной функции F от $n+1$ первого интеграла системы дифференциальных

[1]) Ср. с замечанием относительно модели д'Арси Томсона, следующим после уравнения (22).

уравнений Пфаффа

$$\frac{dx}{\alpha x} = \frac{dy}{\beta y} = \frac{dy'}{(\beta - \alpha)\, y'} = \ldots = \frac{dy^{(n)}}{(\beta - n\alpha)\, y^{(n)}}. \tag{20}$$

Результат определяется выражением

$$F\left[(y/x)^{\gamma},\ y'/x^{\gamma-1},\ \ldots,\ y^{(n)}/x^{\gamma-n}\right] = 0, \tag{21}$$

где $\gamma = \beta/\alpha$. Аргументами F являются однородные степенные функции.

Таким образом, любые деформации в тканях, в которых новообразование определяется степенными функциями, характеризуются действием производной Ли L_d и ее продолжений. Орбиты, определяемые степенными функциями, и соответствующие дифференциальные инварианты обеспечивают произвольные градации сложности новообразований в тканях.

Правдоподобность развития в соответствии со степенными функциями с биологической точки зрения подтверждается явлением аллометрического роста [24]. Преобразования подобия, аналогичные рассмотренным выше, делают возможной генетическую репродукцию — «копирование» — систем органов. Как замечает Стал [24]:

«Представляется вполне правдоподобным, что количественное значение любого биологического критерия подобия может задаваться генетически и прослеживаться в филогенезе. Вероятно, гены переносят наследственную информацию в безразмерной форме, а ее преобразование в переменные, являющиеся размерными величинами, осуществляется ... механизмами, характер которых не совсем ясен...»

Из предшествующего описания следует, что соответствующий переход от генетической информации (параметрической группы) к размерным переменным (многообразие тканей, рассматриваемое как семейство траекторий) оказывается чрезвычайно простым, если его представить как следствие аксиомы 5, т. е. благодаря возможности описывать деятельность клетки с помощью производной Ли.

Хаксли [13] и д'Арси Томсон, изучая явления подобия при новообразовании, использовали понятие «потенциала относительного роста»:

$$\left(\frac{dN_1/N_1}{d\tau}\right)\Big/\left(\frac{dN_2/N_2}{d\tau}\right). \tag{22}$$

В этом выражении переменные N_1 и N_2 представляют, как правило, длины (поверхностный и радиальный рост соответственно по терминологии д'Арси Томсона)[1]), а τ — время. Если выраже-

[1]) Сравните с обсуждением смысла уравнения (17).

ние (22) воспроизводит некоторую локальную константу λ, то оно оказывается эквивалентным системе уравнений Пфаффа

$$\frac{dN_1}{N_1} = \lambda \frac{dN_2}{N_2} = \frac{d\tau}{1}. \tag{23}$$

Эта система (23) в свою очередь характеризует инвариантность относительно действия производной Ли

$$L_a = N_1 \frac{\partial}{\partial N_1} + \frac{N_2}{\lambda} \frac{\partial}{\partial N_2}, \tag{24}$$

эквивалентной при $\gamma = (\beta/\alpha)\ 1/\lambda$ производной Ли, определяемой уравнением (18).

Стал [24] отмечает, что ключевая роль в процессе новообразования принадлежит двум основным принципам: сохранения объема и синхронизации во времени (времена реализации процессов, связанных каким-либо образом, также связаны между собой). Первый принцип оказывается непосредственным следствием второго, если принять во внимание инфинитезимальное преобразование, соответствующее уравнению (23):

$$\delta V = L_a V \cdot \delta\tau, \tag{25}$$

где V — элемент объема, спроецированный в пространство (N_1, N_2). В результате воздействия производной Ли L_a на объемный элемент V в течение временного интервала $\delta\tau$ будет осуществлено инвариантное изменение δV величины V, а также, естественно, и любой орбитальной системы (фибриллы и нитевидные органеллы), порождающей этот объем.

Перейдем к нашей последней, шестой аксиоме. Эта аксиома в сочетании с пятью остальными может, по-видимому, обеспечить достаточную основу для построения универсальной и реалистической математической биологии.

Аксиома 6. *Организм приобретает биологические структуру и функцию, обеспечивающие оптимальность его функционирования с точки зрения трофических функций.*

Так, например, развитие эмбриона морского ежа на ранних стадиях отражает основные типы симметрий, присутствующие в структуре наружного скелета. Однако с наступлением завершающей стадии гаструляции (свободно плавающая личинка) пищеварительный канал зародыша начинает интенсивно сворачиваться в спираль, приобретая существенную асимметрию, с тем чтобы обеспечить максимальную площадь поверхности в пределах заданного объема. Следовательно, «конструкция» пищеварительного канала определяется не столько приспособлением к физическим характеристикам внешней среды, сколько условием максимизации всасываемых питательных веществ.

Аналогичным образом сердечно-сосудистая система высших животных должна быть развита таким образом, чтобы обеспечивать адекватный кислородный метаболизм для других систем органов, возникших в результате эволюционного приспособления к условиям и свойствам внешней среды. Рашевски [19, стр. 66] рассмотрел сердечно-сосудистую систему с точки зрения введенного им же принципа адекватного проектирования, к которому мы также обращаемся в нашем обсуждении. Кроме того, Рашевски утверждает [19], что и минимальная длина кишечного канала у низших животных, и его свертывание в спираль у высших представляют собой проявления того же самого принципа. Принцип адекватного проектирования, являющийся в равной степени и принципом оптимальной структуры (конструкции), заключается в следующем [19, стр. 48]:

«Организм, предназначенный для выполнения определенной совокупности биологических функций, характеризующихся известной интенсивностью, обладает наиболее адекватной (оптимальной) из всех возможных конструкций с точки зрения экономии используемых «материалов» и величины энергетических затрат, необходимых для исполнения предписанных функций».

Совершенно очевидно, что проявления этого принципа относятся к области биоэнергетики, в частности, можно было бы упомянуть трофические функции. Область действия принципа адекватного проектирования должна, однако, и ограничиться биоэнергетикой, ибо в противном случае все животные в конечном счете обладали бы одинаковой морфологией. Как мы убедились, именно приспособление к внешней среде *в целом*, т. е. как физическое, так и гомеостатическое и трофическое, определяет структуру и функцию, адекватные для выживания вида и удовлетворенности индивидуума.

ЛИТЕРАТУРА

1. Alexandroff P. and Hopf H., Topologie, Chelsea, New York, 1965.
2. Berrill N. J., Growth, Development, and Pattern, Freeman, San Francisco, 1961.
3. Birkhoff G. and Rota G.-C., Ordinary Differential Equations, Blaisdell, Waltham, Mass. (1962).
4. Bonner J. T., Morphogenesis, Princeton Univ. Press, Princeton, N. J., 1952.
5. Byers B. and Porter K. R., Oriented Mikrotubules in Elongating Cells of the Developing Cells Rudiment After Induction, *Proc. Nat. Acad. Sci. U. S.*, 52, 1091 (1964).
6. Cohen A., An Introduction to the Lie Theory of One-Parameter Groups, Hafner, New York, 1931.
7. Eisenhart L. P., Continuous Groups of Transformations, Dover, New York, 1961. Русский перевод: Эйзенхарт Л. П., Непрерывные группы преобразований, М., ИЛ, 1947 (перевод с издания 1933 года).
8. Green D. E. and Goldberger R. F., Molecular Insights into the Living Process, Academic, New York, 1967. Русский перевод: Грин Д. Э. и Гольдбергер Р. Ф., Молекулярные аспекты жизни, М., «Мир», 1968.

9. Guggenheimer H. W., Differential Geometry, Mc-Graw-Hill, New York (1963).
10. Hoffman W. C., The Neuron as a Lie Group Germ and a Lie Product, *Quart. Appl. Math.*, **25**, 423 (1968).
11. Holden A. and Bregmann J., Structural and Dynamical Symmetry, *Science Teacher*, **34**, 72 (1967).
12. Holmes R. L., Living Tissues, Pergamon, Oxford, 1965.
13. Huxley J. S., Problems in Relative Growth, Methuen, London, 1932.
14. Jacobson A. G., Inductive Processes in Embryonic Development, *Science*, **152**, 25 (1966).
15. Kobayashi S. and Nomizu K., Foundations of Differential Geometry I, Interscience, New York, 1963.
16. Konigsberg I. R., Clonal Analysis of Myogenesis, *Sciense*, **140**, 1273 (1963).
17. Montgomery D., Compact Groups of Transformations, in Differential Analysis, Oxford Univ. Press, Oxford (1964), p. 43.
18. Racah G., Group Theory and Spectroscopy, Springer tract in modern physics No. 37 (G. Höhler, Ed.), Springer-Verlag, New York (1965), p. 28.
19. Rashevsky N., Mathematical Principles in Biology, Chas. C. Thomas, Springfield, Ill. (1961).
20. Schmitt F. O., Fibrous Proteins — Neuronal Organelles, *Proc. Nat. Acad. Sci. U. S.*, **60**, 1092 (1968).
21. Singer I. M. and Sternberg S., The Infinite Groups of Lie and Cartan, Part I (the Transitive Groups), *J. d'Analuse Math.*, **15**, 1 (1965).
22. Spanier E. H., Algebraic Topology, MsGraw-Hill, New York (1966). Русский перевод: Спеньер Э., Алгебраическая топология, М., «Мир», 1971.
23. Spratt N. T., Introduction to Cell Differentiation, Reinhold, New York (1964).
24. Stahl W. R., Similarity and Dimensional Methods in Biology, *Science*, **137**, 205 (1962).
25. Sternberg S., Lectures on differential Geometry, Prentice-Hall, Englewood Cliffs, N. J. (1964).
26. Thompson D. W., On Growth and Form, Abridged ed., Cambridge Univ. Press, Cambridge (1961).
27. Tondeur P., Introduction to Lie Groups and Transformation Groups, Springer-Verlag, Berlin (1965).
28. Waddington C. H., Principles of Development and Differentiation, Macmillan, New York (1966).
29. Wilmer E. N., Growth and Form in Tissue Culture, in Essays on Growth and Form Presented to D'Arcy Wentworth Thompson (W. L. L. Clark and P. B. Medawar, Eds.), Clarendon, Oxford (1945), p. 264.

СОДЕРЖАНИЕ

КИБЕРНЕТИЧЕСКИЙ СБОРНИК № 12

Редактор А. А. Бряндинская
Художник Н. К. Сапожников Художественный редактор В. И. Шаповалов
Технический редактор Л. П. Чуркина

Сдано в набор 13/II 1975 г. Подписано к печати 15/IX 1975 г. Бумага № 3 60×90¹/₁₆.
6,5 бум. л. 13 усл. печ. л. Уч.-изд. л. 12,74. Изд. № 1/8253. Цена 1 р. 48 к. Зак. 561.

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

Ордена Трудового Красного Знамени Ленинградская типография № 2 имени Евгении Соколовой Союзполиграфпрома при Государственном комитете Совета Министров СССР по делам издательств, полиграфии и книжной торговли.
198052, Ленинград, Л-52, Измайловский пр., 29