ЛОГИЧЕСКИЕ МЕТОДЫ В ИНФОРМАТИКЕ
(Вычислительные системы)
1993 год Выпуск 148

УДК 519.95:681.3

РЕФЛЕКСИВНЫЕ МАШИНЫ ТЬЮРИНГА И ИСЧИСЛЕНИЯ

М.С.Бургин

В в е д е н и е

Если рассматривать логику не в качестве чисто формальной системы, функционирующей внутри математики, а как средство отображения окружающей человека действительности, то окажется, что обычные логические исчисления уже не могут удовлетворить запросы практики. Так, например, в дополнительных комментариях к книге [1] отмечается, что трудности, связанные с открытостью баз знаний, пока не удалось преодолеть чисто логическими средствами, что вызвало разочарование в возможности манипулировать знаниями на уровне обычных логических систем. Более того, если в математике рассматривать такие области, как теория групп, теория вероятностей или теория алгоритмов, так как они реально существуют, то окажется, что рамки логических исчислений (элементарных теорий) являются слишком узкими и нужны более общие структуры для полного представления и изучения этих областей средствами математической логики.

Фиксированность системы аксиом служит слишком сильным ограничением в тех случаях, когда необходимо учитывать динамику. Меняется мир вокруг (или, другими словами, предметная область) и в соответствии с этим должны изменяться аксиомы, посредством которых формализованы основные свойства предметной области. Это приводит к понятию рефлексивного исчисления, которое изучает-

ся в настоящей работе. У такого исчисления в процессе дедукции могут изменяться исходные аксиомы и (или) правила вывода.

В качестве средства изучения рефлексивных исчислений используются рефлексивные машины Тьюринга - алгоритмические устройства, способные изменять свою программу (правила действий головки) в процессе вычислений [2]. Между рефлексивными исчислениями и рефлексивными машинами Тьюринга устанавливаются взаимно-однозначные соответствия. Для них строятся расширяющиеся классы, в основе выделения которых находятся разные уровни рефлексии.

Доказано, что между рефлексивными и обычными машинами Тьюринга можно установить взаимно-однозначное соответствие так, что сопоставленные друг другу машины вычисляют одну и ту же функцию. Из этого вытекает, что класс рефлексивных машин Тьюринга по своей вычислительной мощности равен классу обычных машин Тьюринга. Тем самым опровергается гипотеза С.Клини [3]. В ней предполагается,что,допустив,чтобы программа вычислительного устройства изменялась в процессе вычислений, можно получить более общее, т.е. позволяющее вычислять больше функций,понятие алгоритма по сравнению с обычными машинами Тьюринга, частично рекурсивные функции или другие подобные конструкции алгоритма.

Важным результатом является теорема 3, в которой устанавливается возможность ускорения вычислений в рефлексивных машинах Тьюринга по сравнению с обычными машинами Тьюринга.

Доказательства полученных результатов приводятся, как это принято в последнее время в теории алгоритмов и вычислений, в эскизном виде и не являются полностью формализованными.

В дальнейшем будут использоваться следующие обозначения. Если r - алгоритм, определенный на множестве K и $L \subseteq K$, то $r(L)$ - множество всех результатов применения r к элементам из L. Если H - множество таких алгоритмов, то $H(L) = \bigcup_{r \in H} r(L)$.

1. Исчисления и их свойства

Определяемая конструкция является более общей, чем обычное логическое исчисление. В качестве исходного берется общее понятие исчисления, включающее и логические формальные исчисления.

Пусть $\mathbf{L}$ - некоторый логический язык, или язык выражений, а $\mathbf{R}$ - алгоритмический язык, или язык правил вывода. Язык $\mathbf{L}$ называется логическим, так как он служит основой для построения логических исчислений, хотя во многих случаях он может быть другого типа (как, например, в операционном исчислении или исчислении конечных разностей). Выражения языка $\mathbf{R}$ являются описаниями алгоритмов (правилами), которые преобразуют одни выражения языка в другие. Это означает, что заданы правила применения выражений из $\mathbf{R}$ к выражениям из $\mathbf{L}$. В этом случае правила применения играют роль абстрактной вычислительной машины, выражения из $\mathbf{R}$ - роль программ, а из $\mathbf{L}$ - роль данных. Примером $\mathbf{L}$ может служить язык предикатов первого порядка или вообще любой язык математической логики. В качестве $\mathbf{L}$ можно брать практически любой язык: математический, программирования, химических формул и т.д. В частности, $\mathbf{L}$ может быть и алгоритмическим языком, т.е. языком описания алгоритмов. Примером $\mathbf{R}$ может служить любой язык, на котором описываются правила дедукции.

ОПРЕДЕЛЕНИЕ 1. Исчислением называется тройка (именованное множество [4]) вида $\mathbf{Cal} = (\mathbf{A},\mathbf{H},\mathbf{T})$, где $\mathbf{A},\mathbf{T} \subseteq \mathbf{L}$, $\mathbf{A}$ - базис (система аксиом) исчисления, $\mathbf{H}$ - правила вывода исчисления, т.е. множество алгоритмов из $\mathbf{R}$ вместе с правилами их применения, и $\mathbf{T}$ - тело исчисления $\mathbf{Cal}$, элементы которого получаются применением алгоритмов из $\mathbf{H}$ к выражениям из $\mathbf{A}$.

Многочисленные примеры исчислений можно найти в логике. Это и классические исчисления высказываний и предикатов, и различные модальные и временные логики, и нечеткая логика. Много исчислений существует в математике и помимо логики: матричное, векторное, тензорное, операционное исчисления, классические дифференциальное и интегральное исчисления и т.д.

ОПРЕДЕЛЕНИЕ 2. Исчисление Cal называется:

1) замкнутым по $A(H)$, если все элементы из A подвергаются преобразованиям (все алгоритмы из H используются при порождении T);

2) базисно замкнутым, если любой алгоритм из H применим к любому множеству из A;

3) транзитивно замкнутым, если допускается последовательная композиция любых алгоритмов из H;

4) вполне замкнутым, если оно замкнуто по A, H и транзитивно замкнуто;

5) допустимым, если $T \neq L$;

6) непротиворечивым относительно некоторого множества $P \subseteq L$, если $T \cap P = \emptyset$, где $\emptyset$ - пустое множество;

7) (слабо) полным относительно некоторого множества $F \subseteq R$ и множества $K \subseteq L$, если применение алгоритмов из F к выражениям из T позволяет получить все K, т.е. $K \subseteq F(T)$ (если из $a \in K$ следует либо $a \in F(T)$, либо $\neg a \in F(T)$);

8) непротиворечивым, если оно не содержит формул вида $a \wedge \neg a$. (В этом случае предполагается, что в алфавит языка L входят унарный символ $\neg$ (отрицание) и бинарный символ $\wedge$ (конъюнкция));

9) полным, если для любого $a \in L$ оно содержит a или его отрицание $\neg a$.

Пусть исчисления $Cal_1 = (A_1, H_1, T_1)$, $Cal_2 = (A_2, H_2, T_2)$ вполне замкнуты.

ЛЕММА 1. *Если* $A_1 \subseteq A_2$ *и* $H_1 \subseteq H_2$, *то* $T_1 \subseteq T_2$.

ЛЕММА 2.

а) Если $T \subseteq L$ *и* $H(T) = T$, *то* T *- тело некоторого транзитивно замкнутого исчисления.*

б) Если T *- тело транзитивно замкнутого исчисления, то* $H(T) = T$ *и для любого* r *из* H *выполняется* $r(T) \subseteq T$.

ДОКАЗАТЕЛЬСТВО.

а) Если $H(T) = T$, то положим $Cal = (T,H,T)$. По определению 1 это будет исчисление. Кроме того, так как $H(H(T)) = H(T) = T$, то оно транзитивно замкнуто.

б) Второе утверждение вытекает из определения 2.

Пусть H содержит тождественный оператор.

ЛЕММА 3. *Если* $H(T) \subseteq T$, *то* T *- тело некоторого транзитивно замкнутого исчисления.*

ЛЕММА 4. *Пересечение любого множества тел транзитивно замкнутых исчислений будет телом некоторого транзитивно замкнутого исчисления.*

ДОКАЗАТЕЛЬСТВО. Пусть $Cal_i = (A_i,H_i,T_i)$, $i \in I$, и $T = \bigcap_{i\in I} T_i$. По лемме 2 для всех $i \in I$ имеем $H(T_i) = T_i$. Тогда $H(T) = H(\bigcap_{i\in I} T_i) \subseteq \bigcap_{i\in I} H(T_i) = \bigcap_{i\in I} T_i = T$ и утверждение вытекает из леммы 3.

ЛЕММА 5. *Если* $Cal_1 = (A_1,H_1,T_1)$, $Cal_2 = (A_2,H_2,T_2)$ *и* $H_1 \subseteq H_2$, *то из транзитивной (базисной) замкнутости* Cal_2 *следует транзитивная (базисная) замкнутость* Cal_1.

ОПРЕДЕЛЕНИЕ 3. Системы $H, K \subseteq R$ называются эквивалентными относительно $A \subseteq L$, если $H(A) = K(A)$.

Пусть множество H замкнуто относительно последовательной композиции.

ПРЕДЛОЖЕНИЕ 1. *Исчисление* $Cal = (A,H,T)$ *транзитивно замкнуто тогда и только тогда, когда для некоторого множества* $K \subseteq R$, *эквивалентного* H *относительно* A, *исчисление* $Cal_0 = (A,K,T)$ *замкнуто по* K.

ДОКАЗАТЕЛЬСТВО.1. Пусть исчисление Cal транзитивно замкнуто и K - минимальное подмножество H, эквивалентное ему относительно A. Если какой-то из алгоритмов, принадлежащих K, не применяется к A при порождении T, то его можно исключить из K. Новое множество будет эквивалентно H, хотя оно является собственным подмножеством K. Это противоречит минимальности K, и, следовательно, исчисление Cal_0 замкнуто по K.

2. Если исчисление H замкнуто по K, то в силу эквивалентности K и H относительно A исчисление Cal будет замкнутым по H. Так как множество H замкнуто относительно последовательной композиции алгоритмов, то это означает транзитивную замкнутость Cal (Cal_0 по лемме 5).

ОПРЕДЕЛЕНИЕ 4. Исчисления $Cal_1 = (A_1,H_1,T_1)$ и $Cal_2 = (A_2,H_2,T_2)$ называются эквивалентными, если $T_1 = T_2$.

ПРЕДЛОЖЕНИЕ 2. *Свойства 6-10 из определения 2 сохраняются для эквивалентных исчислений, а свойства 1-5 - необязательно.*

ЗАМЕЧАНИЕ 1. Понятие эквивалентности особенно важно для приложений, так как во многих случаях большое значение имеет эффективность работы с исчислением. Многие результаты становятся недостижимыми, если вывод осуществляется слишком медленно. Например, если проблемная область D некоторой экспертной системы описана с помощью системы продукций A, то A играет роль системы аксиом (базиса) исчисления. Система A может хорошо (т.е. достаточно полно и точно) описывать область D, но быть неудобной для обработки. В такой ситуации целесообраз-

но найти другую систему продукций (аксиом) B, эквивалентную A, но существенно более эффективную. О том, насколько скорость обработки информации на основе логического исчисления зависит от выбора системы аксиом, свидетельствует теорема А.Эренфойхта и Я.Мыцельского об ускорении вывода в формальных системах (см. [3]).

Аналогичная зависимость от правил (алгоритмов) вывода демонстрируется теоремами М.Рабина и М.Блюма (см.[4]). При этом такое ускорение может быть получено не только для обычных мер сложности вычислений (как в [4]), но и для двойственных к ним (см. [5]).

Как отмечает специалист в области автоматического доказательства теорем С.Ю.Маслов (см.[6]), основным средством сокращения выводов в исчислениях является введение дополнительных дедуктивных средств - новых правил вывода.

Все эти соображения, а также предположения о возможном изменении предметной области, которая описывается исчислением, приводит к необходимости изменять исчисление во время работы с ним. Могут меняться исходные аксиомы, могут трансформироваться допустимые правила вывода. Строгие математические конструкции, позволяющие все это осуществлять, вводятся ниже.

ОПРЕДЕЛЕНИЕ 5. Исчисление $Cal = (A,H,T)$ над языком L называется:

1) тактовым, если выполнены условия:

а) $H = \{H_n ; n = 1,2,\dots\}$;

б) вывод делится на шаги (такты);

в) на шаге n алгоритмы из H_n применяются к T_{n-1};

г) $T = A$;

д) $T_n = H_n(T_{n-1})$;

е) $T = \bigcup_{n=1}^{\infty} T_n$;

2) рефлексивным, если оно тактовое и выполнены условия:

а) $R \subseteq L$;

б) $H_{n+1} = H_n(H_{n-1})$;

3) динамическим, если выполнены условия:

а) $H = H_A \cup H_T$;

б) $T_n = H_T(A_{n-1})$;

в) $A_n = H_A(A_{n-1})$;

4) стратифицированным, если выполнены условия:

а) $A = \bigcup_{k \in K} A_k$;

б) $H = \bigcup_{k \in K} H_k$;

в) $T_k = H_k(A_k)$;

г) $T = \bigcup_{k \in K} T_k$.

ОПРЕДЕЛЕНИЕ 6. Правило r из H называется независимым относительно множества $K \subseteq L$, если из $r(K) \subseteq K$ следует $r(K) \not\subseteq K \setminus H_1(K)$, где $H_1 = H \setminus \{r\}$, независимым в тактовом исчислении $Cal = (A,H,T)$, если r независимо относительно всех множеств T_n , начиная с некоторого n .

Отметим, что одно правило всегда независимо относительно любого множества.

Пусть в тактовом исчислении $Cal = (A,H,T)$ все правила независимы.

ПРЕДЛОЖЕНИЕ 3. *Тактовое исчисление* Cal *будет транзитивно замкнутым тогда и только тогда, когда для любого правила* r *из* H *, начиная с некоторого* n, *имеет место равенство* $r(T_n) \subseteq T$, *либо* r *принадлежит бесконечно многим наборам* H_n .

ДОКАЗАТЕЛЬСТВО. Действительно, по лемме 2 для транзитивности Cal необходимо и достаточно, чтобы для любого алгоритма (правила) r из H выполнялось включение $r(T) \subseteq T$.

Так как $T = \bigcup_{n=1}^{\infty} T_n$, то либо, начиная с некоторого n, имеет место включение $r(T_n) \subseteq T_n$, либо существует бесконечно много номеров $m_1, m_2, \ldots$, для которых $r(T_{m_i}) \not\subseteq T_{m_i}$.

Во втором случае существуют такие элементы $t_1, t_2, \ldots$, для которых имеет место $t_i \in r(T_{m_i}) \setminus T_{m_i}$. Так как исчисление Cal транзитивно замкнуто, то все t_i принадлежат T. А это в силу независимости r возможно только в том случае, когда r входит в некоторую систему правил H_m при $m \geq m_i$. Так как таких номеров m_i бесконечно много, то и r будет принадлежать бесконечно многим наборам H_n. Предложение 3 доказано.

2. Общее представление алгоритмов и их эквивалентности

В дальнейшем понадобится достаточно общее понятие эквивалентности алгоритмов, которое будет рассмотрено ниже.

Обычно определение алгоритмов того или иного вида производится следующим образом. Сначала задаются (или считаются априори заданными) два конструктивных (в некотором смысле) множества X и Y. Первое содержит входные данные алгоритмов и называется их областью определения. Второе содержит результаты (выходные данные) и называется областью значений. Множества X и Y могут совпадать. Например, для частично рекурсивных функций - это множество натуральных чисел, а для машины Тьюринга - символьные последовательности.

Далее, выделяется некоторый класс E элементарных алгоритмов (правил) преобразования элементов из X, а также класс J (правил) их применения и композиции. В некоторых случаях с классом алгоритмов связывается некоторое устройство, работа которого описывается (определяется) алгоритмом. Примерами эле -

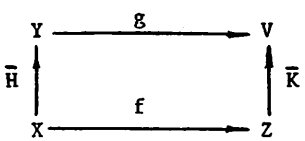
ментарных алгоритмов могут служить команды машин Тьюринга, продукции Поста $x \to y$ или продукции, которые используются в экспертных системах.

Таким образом, каждый алгоритм A - это тройка (именованное множество [4]), имеющая вид (X,H,Y), где H состоит из некоторых правил, принадлежащих E, и правил их применения из J. Система H определяет некоторое частичное отображение $\bar{H}$ из X в Y, которое называется алгоритмическим и сопоставляется алгоритму A. При этом применение алгоритма A к элементу x из X обозначается $A(x)$ и равно $\bar{H}(x)$. В общем случае $A(x)$ не всегда существует.

Для определения алгоритма необходимо описать, что считается результатом его применения, т.е. задать отображение H. Одни и те же правила применения и элементарные алгоритмы могут порождать существенно разные классы алгоритмов в зависимости от того, как определяется результат. Примером этого могут служить классы обычных и индуктивных машин Тьюринга [11].

Пусть F - некоторый класс взаимно-однозначных отображений (биекций).

ОПРЕДЕЛЕНИЕ 7. Алгоритмы (исчисления) $A = (X,H,Y)$ и $B = (Z,K,V)$ называются $\mathcal{F}$-эквивалентными, если существуют биекции $f: X \to Z$ и $g: Y \to V$, для которых $\bar{H}g = f\bar{K}$, т.е. коммутативна следующая диаграмма:



Это означает, что $\mathcal{F}$-эквивалентность является морфизмом именованных множеств, задающих их эквивалентность [4]. В свою очередь, такая эквивалентность может быть получена как ча-

стный случай эквивалентности многомерных операторов, введенной в [11].

Такого рода эквивалентности широко используются в теории алгоритмов, например, при доказательстве такого важного факта, как эквивалентность различных моделей рекурсивных алгоритмов: частично рекурсивных функций, машин Тьюринга, формальных исчислений и т.п.

Эквивалентность исчислений, введенная в определении 4, превращается в $\mathcal{F}$-эквивалентность следующим образом. В качестве X,Y,Z,V берется множество PL всех подмножеств языка L. Если задано исчисление $Cal=(A,H,T)$, то отображение H сопоставляет множеству A множество T и не определено на других подмножествах. В качестве f берется произвольное отображение PL в себя, которое переводит систему аксиом первого исчисления в систему аксиом второго, а в качестве g - тождественное отображение PL на себя.

3. Рефлексивные машины Тьюринга

Для изучения свойств рефлексивных исчислений используются рефлексивные машины Тьюринга, которые бывают разных порядков. Рефлексивная машина Тьюринга порядка 0 - это обычная однолен - точная машина Тьюринга.

При $n > 0$ рефлексивная машина Тьюринга порядка n имеет $n+1$ ленту. Все ленты такие же, как у обычных машин Тьюринга. Одна лента (первая) является рабочей, на ней записываются отрабатываемые данные. Головка, которая с ними работа - ет, называется рабочей и управляется программой (набором команд или инструкций), записанной на второй ленте. Если $n > 1$, то на второй ленте также находится (рефлексивная) головка, которая преобразует программу первой головки по программе, записанной на третьей ленте и т.д. На ленте с номером n, где за-

писана программа головки с номером по $n-1$, работает (рефлексивная) головка с номером n по программе, записанной на ленте с номером $n+1$. На ней находится последняя рефлексивная головка.

Работа рефлексивной машины Тьюринга T происходит следующим образом. На рабочей ленте записывается некоторое слово в алфавите данной машины и T начинает работу из своего начального состояния, в котором рабочая головка находится в ячейке рабочей ленты, содержащей первый символ исходного слова.

Для удобства рассмотрений можно считать, что на каждом шаге работает только одна головка. Для этого в системе команд (правил работы) каждой головки имеются команды передачи управления каждой из остальных головок этой машины. Например, передача управления реализуется по счетчику. На работу каждой головки h_i выделяется определенное число тактов (шагов) r_i. Как только головка h_i начинает работать, счетчик отсчитывает r_i срабатываний головки h_i, а затем при $i < n$ передает управление головке h_{i+1}, которая начинает работать. При $i = n$ происходит передача управления первой головке. Такие машины будем называть счетчиковыми.

Рассмотрим соотношение между обычными и рефлексивными машинами Тьюринга. В докладе на Международном математическом конгрессе в связи с обсуждением тезиса Черча-Тьюринга С.Клини высказал [3] гипотезу о том, что опровергающий пример тезиса Черча должен был бы включать процедуры вычисления, в которых процедура как-то меняется в зависимости от аргумента (обрабатываемых данных).

Следующий результат показывает, что эта гипотеза неверна.

ТЕОРЕМА 1. *Для любой рефлексивной машины Тьюринга* M *порядка* n *существует эквивалентная ей рефлексивная машина Тьюринга* T *порядка* $n-1$, *т.е. машина* T *вычисляет то же множество слов, что и машина* M.

ДОКАЗАТЕЛЬСТВО. Рассмотрим некоторую рефлексивную машину Тьюринга M порядка n. Для построения машины T воспользуемся конструкцией универсальной машины Тьюринга. В теории алгоритмов установлено существование универсальных машин Тьюринга. Устроена универсальная машина U может быть так [10]. На ленте U выделена одна конечная и две бесконечные части. На конечной части записывается текущее состояние моделируемой с помощью U машины Тьюринга Q. На одной бесконечной части содержится запись команд мащины Q в кодах машины U, а вторая бесконечная часть служит для представления рабочей ленты машины Q. Головка машины U, просмотрев текущее состояние Q, ищет подходящую команду, а затем изменяет содержимое части, представляющей ленту Q. После этого (или до этого) изменяется и текущее состояние Q в соответствии с выбранной командой.

Внесем изменение в систему команд C машины U так, чтобы она не только просматривала на своей ленте запись команд машины T, но и меняла их в соответствии с программой рефлексивной машины M по изменению программы работы $(n+1)$-й головки M, находящейся на $(n+1)$-й ленте. Это можно сделать, так как программа рабочей головки M содержит обычные команды машин Тьюринга. Свойства полученной из C с помощью таких изменений новой системы команд B удовлетворяют всем требованиям к системам команд машин Тьюринга.

Далее, удалим в машине M $(n+1)$-ю ленту, а n-ю ленту заменим на ленту машины M. Кроме того, в качестве системы команд головки, которая находится на этой ленте, возьмем R.

Свойства универсальной машины Тьюринга U показывают, что построенная рефлексивная машина Тьюринга T эквивалентна исходной машине M и имеет порядок $n-1$.

СЛЕДСТВИЕ 1. *Для любой рефлексивной машины Тьюринга существует эквивалентная ей обычная машина Тьюринга.*

ДОКАЗАТЕЛЬСТВО получается применением теоремы 1 нужное число раз.

СЛЕДСТВИЕ 2. *Для любых натуральных чисел* **n** *и* **m** *мощности классов рефлексивных машин Тьюринга порядка* **n** *и* **m** *совпадают и равны мощности любого класса рекурсивных алгоритмов типа обычных машин Тьюринга.*

ТЕОРЕМА 2. *Между рефлексивными исчислениями, заданными с помощью продукции Поста, и рефлексивными исчислениями существует взаимно-однозначное соответствие, которое каждому исчислению сопоставляет эквивалентную ему машину.*

Доказательство проводится стандартным образом так же,как и для обычных исчислений и машин Тьюринга.

СЛЕДСТВИЕ 1. *Для любого рефлексивного исчисления существует эквивалентное ему обычное исчисление.*

СЛЕДСТВИЕ 2. *Мощность класса рефлексивных исчислений, заданных системами продукций, совпадает с мощностью класса обычных исчислений, заданных системами продукций, которая равна мощности любого универсального класса рекурсивных алгоритмов* (например,машин Тьюринга).

ЗАМЕЧАНИЕ 2. Интересной представляется задача рассмотреть механизмы рефлексии, аналогичные введенным выше, для суперрекурсивных алгоритмов и, в частности, для индуктивных машин Тьюринга [9].

4. Ускорение в рефлексивных машинах Тьюринга

Введение рефлексивных машин Тьюринга оправдано уже тем, что они позволяют реагировать на внешние изменения преобразованием своих программ. В то же время, как было показано выше, класс вычислимых функций они не расширяют. Поэтому возникает вопрос, может быть,они будут более эффективными, чем обычные

машины Тьюринга? Следующие результаты о возможности ускорения вычислений показывают, что это действительно так.

Прежде чем к ним переходить, заметим, что ускорение естественно рассматривать только для машин с большой временной сложностью вычислений. Если такая сложность невелика, то процесс и без этого реализуется достаточно эффективно.

Напомним, что временная сложность (алгоритмов) машины Тьюринга T - это функция $\alpha_T(n)$ $(\beta_T(n))$, значение которой для натурального числа n равно максимальному (минимальному) числу шагов (времени), которые делает T, начиная работу со словом длины n, а если такого максимума (минимума) не существует, то $\alpha_T(n)$ $(\beta_T(n))$ не определено. Первая из этих мер α_T оценивает время вычислений сверху, рассматривая наихудший случай, а вторая α_T - снизу, рассматривая наилучший случай. Обычно (см., например, [12]) рассматриваются только разрешимые проблемы, которым соответствуют всюду определенные машины Тьюринга, а сложность определяется по максимуму.

Будем предполагать (условие Р3), что $\alpha_T(n) \geq n^3$. Это означает существование такого n_0, что $\alpha_T(n) > n^3$ при $n \geq n_0$. Таких машин достаточно много, так как в их число входят машины с экспоненциальным временем вычислений. Кроме того, все рассматриваемые машины Тьюринга могут работать с бесконечным алфавитом, каждый символ которого может быть помещен в ячейку или удален из нее (заменен) за один такт работы машины.

ТЕОРЕМА 3. *Для любой машины Тьюринга* T, *удовлетворяющей условию* (Р3), *можно построить (конструктивно определить) рефлексивную машину Тьюринга* M *первого порядка, для которой при любом натуральном числе* t *выполняется соотношение* $\alpha_M(n) \prec \frac{1}{t}\alpha_T(n)$,

т.е., начиная с некоторого m *при* $n \geq m$, *имеет место неравенство* $\alpha_M(n) < \frac{1}{t}\alpha_T(n)$.

ДОКАЗАТЕЛЬСТВО. Известно [13], что любой машине Тьюринга T_1 можно поставить в соответствие машину Тьюринга T_2, эквивалентную T_1, но работающую в n раз быстрее, где n - любое натуральное число. Делается это за счет более сжатого кодирования информации в машине T_2 . Идея доказательства теоремы 3 заключается в том, что рефлексивная машина может регулярно перекодировать как обрабатываемые данные, так и правила их обработки. Нужно только, чтобы время, затрачиваемое на эти перекодировки, компенсировалось ускорением за счет сжатия информации. Покажем, как это можно сделать.

Нужная машина Тьюринга M строится как композиция более простых машин Тьюринга T_c, T_p, T_{kd}, T_{kr} . Машина T_c осуществляет подсчет символов, входящих в исходное слово x_0, записанное на рабочей ленте. Машина T_p вычисляет, сколько раз нужно производить кодирование, с помощью которого осуществляется сжатие. Машина T_{kd} осуществляет кодирование слова x, которое в момент начала ее работы находится на рабочей ленте. Машина T_{kr} делает то же самое с набором правил, по которым функционирует рабочая головка машины M . Из инструкций (правил) машин T_c, T_p и T_{kd} и исходной машины T при композиции строится система правил рабочей головки машины M . Правила машины T_{kr} используются для функционирования рефлексивной головки машины M .

Композиция производится следующим образом. Сначала параллельно работают машины T_c и T_p, причем вначале выполняет одно действие T_c, потом - одно действие T_p, потом - снова T_c и так далее. Когда T_c заканчивает пересчет символов в слове x_0, останавливается и T_p, а ее результат, указывающий нужное число перекодирований, записывается на обе ленты машины M . После этого, если число перекодировок боль-

ше нуля, срабатывает каждая из машин T_{kd} и T_{kr}. Очередность их работы произвольна, так как они независимо работают с разными лентами, а число срабатываний определяется результатом машины T_p. После завершения работы машин T_{kd} и T_{kr} рабочая головка машины M начинает работать по программе, полученной с помощью T_{kr}, с данными, закодированными с помощью T_{kd}.

Один цикл кодирования информации с помощью T заменяет каждую пару символов в x на один новый символ, начиная с левого края x. Пара символов $\{s_i, s_j\}$ кодируется одним символом $s_{k\,i+j}$, где k больше числа r_T команд машины Тьюринга T и количества символов в рабочем алфавите машины T. Подобное перекодирование осуществляется и с правилами так, что каждое новое правило определяет преобразование пары символов, заданное двумя старыми правилами. Описание таких правил содержится в доказательстве теоремы 2 из [13].

Оценим количество шагов $s(x)$, которые при подаче слова длины n на вход машины M делаются машинами T_c, T_p, T_{kd}, T_{kr} при одном их срабатывании. Так как машины T_c и T_p работают параллельно и вместе заканчивают работу, то они делают $s(T_c) + s(T_p) = C_0 n$ шагов, где C_0 - некоторая константа. Одна перекодировка слова длины n, учитывая сдвиги головки машины Тьюринга T_{kd}, требует не более $C_1 n^2$ шагов, где C_1 - константа. Если количество перекодировок равно p, то T_{kd} делает не более чем $pC_1 n^2$ шагов, так как длина кодируемого слова не увеличивается.

При кодировании правил машины Тьюринга T с помощью машины Тьюринга T_{kr} каждому правилу машины T ставится в соответствие не более $2m_T^3$ правил машины M (см. [13]). Длина каждого нового правила больше, чем длина исходного и их от-

ношение оценивается константой C_2. Поэтому длина нового набора правил не превосходит $2C_2 r_T m_T^3 = n_T$, а число шагов для их порождения оценивается величиной $C_3 n_T^2$, где C_3 не зависит от T.

На втором цикле кодирования правил число r_{T_2} кодируемых правил возрастает в пределах $2r_T m_T^3$, а число новых правил, поставленных в соответствие одному старому, оценивается той же константой $2m_T^3$. Поэтому длина набора правил, получаемых на втором цикле кодирования, не превосходит $2C_2 r_T m_T^6 = n_{T_2} \leq$ $\leq n_T^2$. Число шагов машины T_{kr}, нужных для порождения этих правил, оценивается величиной $C_3 n_T^4$. Продолжив эти рассмотрения, получим, что p кодирований потребуют не более чем $C_3(n_T^2 + n_T^4 + \ldots + n_T^{2p}) \leq C_4 n_T^{2p}$ шагов, где C_4 - некоторая компонента.

Таким образом, все преобразования займут не более $n_{T_r} =$ $= C_0 n + pC_1 n^2 + C_4 n_T^{2p} \leq C_5(pn^2 + (n_T^p)^2)$ шагов.

Предположим, что $p \leq \log_a n$ для некоторого натурального числа a. Тогда

$$n_{T_r} \leq C_5(n^2 \log_a n + b_T^{\log_a n}),$$

где $b_T = n_T^2$. Так как

$$\log_a n = \frac{\log_{b_T} n}{\log_{b_T} a},$$

то

$$b_T^{\log_a n} = \left(b_T^{\log_{b_T} n}\right)^{\frac{1}{\log_{b_T} a}} = n^{\frac{1}{\log_{b_T} a}} .$$

Если $a > b_T$, то $n^{\frac{1}{\log_{b_T} a}} < n$ и имеем $n_{T_r} \le Cn^2 \log_a n$ для достаточно больших n.

Выберем такое натуральное число n_0, для которого имеют место неравенства $n_0 > b_T$, $n_0^3 > 4C(n_0^2 \log n_0 + n_T^2)$. Это можно сделать, так как C и n_T не зависят от n и $\lim\limits_{n\to\infty} \frac{n}{\log n} = \infty$.

Опишем теперь, как работает машина Тьюринга T_p. Имея на входе запись числа n_0 (на рефлексивной ленте машины M), она последовательно вычисляет $n_1 = n_0^3$, $n_2 = n_1^3$ и так далее. После того, как вычислено очередное n_p, в счетчик числа перекодирований записывается число p. В начале там стоит 0. Содержимое счетчика дает входную информацию для машин T_{kd} и T_{kr}, указывая, сколько раз нужно производить перекодирование. При этом $n_p = n_0^{3^p}$, из чего $3^p \log n_0 < \log n$ и

$$p < \frac{1}{3} \frac{\log n}{\log n_0} = \frac{1}{3} \log_{n_0} n < \log_{n_0} n .$$

В то же время, в силу выбора числа n_0 имеем $n_0 > b_T$, т.е. выполнены те условия, которые ранее предполагались истинными при выведении неравенств.

Вычисление куба натурального числа x реализуется машиной Тьюринга и требует выполнения не меньшего числа операций, чем x^3 [14]. Поэтому, если машина T_p показала, что нуж-

на одна перекодировка, то из этого следует, что $n > n_0^3$. Тогда машина M, прежде чем начать обрабатывать входное слово a, выполнит перекодировку. Затем она на обработку слова a затратит $\frac{3}{2} t_T(a)$ операций, где $t_T(a)$ – число шагов машины Тьюринга T в ходе обработки слова a. На операции по перекодировке затрачено, как было показано, не более $Cn^2 \log n$ шагов. Если слово a определяет значение функции α_T, т.е. $\alpha_T(a) = t_T(a)$, то $t_T(a) \geq n^3 > 4Cn^2 \log n$, так как $n > n_0$ и $f(n) = n^3 - 4Cn^2 \log n$ – монотонная функция при $n > 4C \log n$. Тогда общее число операций машины M, затраченных на обработку слова a, не будет превосходить

$$\frac{1}{2} t_T(a) + Cn^2 \log n < \frac{1}{2} t_T(a) + \frac{1}{4} t_T(a) = \frac{3}{4} t_T(a).$$

Если нужно выполнить две перекодировки, то $n > n_1^3 = (n_0^3)^3$. Тогда $(n)^3 - (4Cn^2 \log n)^3 > (n_0^3)^3 - (4Cn_0^2 \log n_0)^3 > n_0^3 - 4Cn_0^2 \log n_0 > 0$ по условию. Отсюда $n^3 > 64C^3 n^6 \log^3 n > 16\, Cn^2 \log n$ и $Cn^2 \log n < \frac{1}{16} n^3$. Значит, вся обработка слова a потребует не более $\frac{1}{4} t_T(a) + \frac{1}{16} t_T(a) = \frac{3}{16} t_T(a)$ шагов.

При p перекодировках число шагов рефлексивной машины Тьюринга M на входе a не будет превосходить $\frac{3}{2^{2p}} t_T(a)$. Выберем такое p, чтобы выполнялось неравенство $2^{2p} > 3t$, где t – число, заданное в условии теоремы. Тогда время $t_M(a)$ обработки слова a машиной M будет меньше, чем $\frac{1}{t} t_T(a)$, а результат обработки будет таким же.

Кроме того, для сложности $\beta_M(n)$ имеем $\beta_M(n) \leq$ $\leq t_M(a) < \frac{1}{t} t_T(a) = \frac{1}{t}\beta_T(a)$. По определению $\alpha_M(a) = t_M(b)$ для некоторого слова b. Тогда по построению $\alpha_M(n) = t_M(b) < \frac{1}{t}t_T(b) \leq \frac{1}{t} t_T(a) = \frac{1}{t}\alpha_T(n)$. Так как машины Тьюринга T и M эквивалентны, то теорема доказана.

ЗАМЕЧАНИЕ 4. Полученный результат показывает, что если для ускорения вычислений в классе обычных машин Тьюринга нужно каждый раз менять машину на более быструю, то в классе рефлексивных машин Тьюринга ускорять вычисления можно на одной и той же машине. Более того, такая рефлексивная машина Тьюринга позволяет достичь бóльшего ускорения, чем любое конечное число замен обычных машин Тьюринга.

ЗАМЕЧАНИЕ 5. Теорема была доказана в предположении, что в рассматриваемом классе машин Тьюринга возможна работа с бесконечными алфавитами, а число состояний этих машин неограниченно. Однако и в случае машин Тьюринга с такими ограничениями рефлексивность может повысить эффективность работы за счет того, что слова небольшой длины (данные небольшого объема) обрабатываются прямо, а ускорение, требующее дополнительных операций по перекодированию, применяется только к достаточно длинным словам. Особенно большой выигрыш достигается для алгоритмов, имеющих экспоненциальную временную сложность.

ЗАМЕЧАНИЕ 6. Результат, доказанный в теореме 3, остается справедливым и для временной меры сложности, которая изучалась в [13] и отлична от мер сложности, рассмотренных выше.

ЗАМЕЧАНИЕ 7. Интересно было бы изучить возможности уменьшения с помощью рефлексивных машин Тьюринга сложности вычислений для других мер сложности. Примерами таких мер являются ленточная, поворотная и другие прямые меры сложности [7, 12].

Заключение

Многие приложения логики (экспертные системы с динамическими предметными областями, модели научных теорий, теория практических рассуждений и аргументации) обнаруживают недостаточность обычных логических средств (исчислений, стандартных моделей алгоритмов). С целью преодоления этих недостатков введены рефлексивные исчисления и логические многообразия, индуктивные и рефлексивные машины Тьюринга. В работе показано, что хотя применение рефлексивных машин Тьюринга позволяет значительно повысить эффективность обработки информации, интересной является задача введения рефлексивных механизмов в другие модели алгоритмов: в вероятностные и недетерминированные машины Тьюринга, равнодоступные адресные машины (RAM), машины Минского, алгоритмы Колмогорова и т.д. Рефлексивные механизмы в разных системах приводят к разным эффектам.

Литература

1. КАНДРАШИНА Е.Ю., ЛИТВИНЦЕВА Л.В., ПОСПЕЛОВ Д.А. Представление знаний о времени и пространстве в интеллектуальных системах. - М.: Наука, 1989. - 328 с.

2. BURGIN M.S. On a conjecture of S.Kleene //Abstracts presented to the American Mathematical Society. - 1983.-Vol.4, N 6. - P. 483.

3. КЛИНИ С. Математическая логика. Конструктивные и неконструктивные операции //Международный математический конгресс в Эдинбурге, 1958 г. (Обзорные доклады). - М., 1962. - С. 58-180.

4. БУРГИН М.С. Функторная семантика в категориях именованных множеств //Рациональность, рассуждения, коммуникация. - Киев: Наукова думка, 1987. - С. 183-197.

5. ЭРЕНФОЙГТ А., МЫЦЕЛЬСКИЙ Я. Сокращение доказательств при добавлении новых аксиом //Сложность вычислений и алгоритмов.-М.: Мир, 1974. - С. 172-173.

6. БЛЮМ М. Машинно независимая теория сложности рекурсивных функций //Проблемы математической логики. - М.: Мир, 1970. - С. 401-422.

7. БУРГИН М.С. Обобщенные меры сложности Колмогорова и двойственность в теории вычислений //Докл. АН СССР. - 1982. - Т. 264, № 1. - С. 19-23.

8. МАСЛОВ С.Ю. Ассиметрия познавательных механизмов и ее следствия //Семиотика и информатика. - 1983. - Вып. 20. -С. 3-31.

9. БУРГИН М.С. Индуктивные машины Тьюринга //Докл.АН СССР. - 1983. - Т. 250, № 5. -С. 1289-1293.

10. МИНСКИЙ М. Вычисления и автоматы. - М.: Мир, 1971. - 364 с.

11. БУРГИН М.С. Функциональная эквивалентность операторов и параллельные вычислительные процессы //Программирование.-1980. - № 6. - С. 3-16.

12. АХО А., ХОПКРОФТ Дж., УЛЬМАН Дж. Построение и анализ вычислительных алгоритмов. - М.: Мир, 1979. - 536 с.

13. ХАРТМАНИС Дж., СТИРНЗ Р. О вычислительной сложности алгоритмов //Киб. сб., 1974. - Вып. 11. - С. 131-176.

14. МАЛЬЦЕВ А.И. Алгоритмы и рекурсивные функции. - М.: Наука, 1965. - 392 с.

Поступила в ред.-изд.отд.
18 февраля 1993 года