Записки научных
семинаров ПОМИ
Том 277, 2001 г.

**А. П. Бельтюков**

# КОМБИНИРОВАННЫЕ ИЕРАРХИИ КОНЕЧНЫХ РАВНОДОСТУПНЫХ АДРЕСНЫХ МАШИН

## Введение

Настоящая работа относится к теории сложности вычислений для конечных вычислительных машин на относительно коротких входных данных. Таким образом, в ней реализуется финитарный подход к теории сложности вычислений. Рассматриваются равнодоступные адресные машины, программы и входные данные которых хранятся в памяти. Строятся комбинированные зонно-временные иерархии классов предикатов, распознаваемых этими машинами.

Большинство традиционных результатов теории сложности вычислений (таких, например, как классические работы [1] и [2]) имеют асимптотический характер. По этой причине зачастую их практическое применение затруднено: аргументы, для которых начинают выполняются сложностные ограничения, и коэффициенты в выражениях для верхних границ могут быть слишком велики для того, чтобы иметь реальный смысл. Один из подходов, позволяющих приблизить теоретические результаты к реальности – отказ от асимптотики. При этом рассматриваются не бесконечные машины, а классы конечных машин возрастающей сложности. Реализации этого подхода и посвящена настоящая работа.

Основные отличия рассматриваемой здесь модели от традиционных конечных автоматов и преобразователей – ограничение размера входных данных и учет размера программы. Рассматриваемые в настояще работе машины работают в режиме “off-line” в отличие от конечных автоматов и преобразователей, работающих со сколь угодно большими входами в режиме “on-line” посимвольно. Поэтому для традиционных конечных автоматов и преобразователей теоремы об иерархии по объему памяти (чи-

Работа была частично поддержана грантами РФФИ No. 98-96-80205 и РГНФ No. 98-03-04368.

слу состояний) не очень интересны, так как получаются почти тривиально их мощностных соображений при любом увеличении числа состояний.

Другая близкая модель – схемы из логических элементов. Основное отличие рассматриваемой модели от логических схем – полное отсутствие у последних программы, не зависящей от размера схемы. Практически это означает, что схему нужно разрабатывать заново для каждого размера входных данных. Поскольку схемой из логических элементов можно промоделировать работу любого конечного компьютера, то результат настоящей работы можно сформулировать и на языке схем, но, в силу отмеченной особенности, он на этом языке выглядит весьма искусственно. Такая же иерархия для схем, как и результат настоящей работы, представляется невозможной, так как здесь нельзя ограничить размер программы величиной меньшей, чем размер схемы.

## 1. Основные определения

*Конечная равнодоступная адресная машина* размера $N$ (RAM-$N$) состоит из конечной последовательности регистров (ячеек):

$$r_0, r_1, \dots, r_{N-1}.$$

Работа машины осуществляется в некоторые дискретные моменты времени, которые также образуют конечную последовательность. В каждый момент времени в каждом регистре находится натуральное число в диапазоне от 0 до $N-1$. Заметим, что совпадение здесь диапазона адресов и чисел выбрано нами просто из соображений удобства: рассмотрение двух различных диапазонов ведет к серьезному усложнению построений, не давая никаких радикально новых результатов. В строящейся конструкции легко моделируются существенные черты современных вычислительных машин.

Программа машины представляет из себя конечную последовательность чисел, помещающихся в последовательных начальных адресах памяти (с регистра $r_0$). Способ интерпретации (выполнения и завершения работы) программы описан ниже. Непосредственно вслед за программой в последовательных адресах размещаются входные данные, которые также представляют из себя конечную последовательность чисел в диапазоне от 0 до

$N-1$ (нетрудно видеть, что если на практике потребуются данные в другой форме, то их легко закодировать в виде такой последовательности). Результат работы находится в регистре $r_0$ (аккумуляторе) после завершения работы программы (при распознавании предикатов это 1 или 0, т. е. "да" или "нет" соответственно). То есть нетривиальная программа модифицирует себя. Считается, что во время выполнения программы в регистре $r_1$ находится *адрес выполняемой команды* – некоторое число $i$ в диапазоне от 2 до $N-2$ (при выходе за пределы этого диапазона работа завершается). Значит, в начале работы в этой ячейке находится адрес первой выполняемой команды. Содержимое пары регистров $r_i$ и $r_{i+1}$ считается *выполняемой командой*. Регистр $r_i$ содержит *код выполняемой операции*, а регистр $r_{i+1}$ – *адрес операнда*, т. е. номер регистра, с содержимым которого должна быть произведена операция, обозначенная указанным кодом. Этот номер будем обозначать далее буквой $x$. Способ кодировки операций не очень существенен, но для определенности зафиксируем следующие коды:

**Код 1** – команда LOAD – копирование содержимого $r_x$ в аккумулятор $r_0$,

**Код 2** – команда STORE – копирование содержимого аккумулятора $r_0$ в регистр $r_x$,

**Код 3** – команда ADD – сложение содержимого $r_x$ с содержимым аккумулятора $r_0$: $r_0 := r_0 + r_x$,

**Код 4** – команда SUBT – вычитание содержимого $r_x$ из содержимого аккумулятора $r_0$: $r_0 := \max\{r_0 - r_x, 0\}$ ("арифметическая" – неотрицательная разность),

**Код 5** – команда JZERO – условный переход по адресу операнда при ненулевом содержимом аккумулятора $r_0$ (номером следующей выполняемой команды (в $r_1$) становится $x$):

$$\textbf{if } r_0 = 0 \textbf{ then } r_1 := x \textbf{ else } r_1 := r_1 + 2$$

(заметим, что альтернатива $r_1 := r_1 + 2$ выполняется для всех предыдущих команд автоматически всегда после выполнения основного действия),

**Код 6** – команда JUMP – безусловный переход по адресу операнда (номером следующей выполняемой команды становится $x$): $r_1 := x$.

Никакие команды с непосредственной и косвенной адресацией для подобия алгоритмической полноты не нужны, так как программа может содержать константы и модифицировать себя. Например, для помещения в аккумулятор содержимого регистра, номер которого содержится в регистре номер 3, достаточно с регистра номер $i$ по регистр номер $i+5$ записать команды:

LOAD 3
STORE $i+5$
LOAD 0

При этом вторая команда изменит адрес операнда третей команды (в регистре номер $i+5$) так, чтобы та произвела выборку необходимого числа.

Любая команда с кодом за пределами диапазона от 1 до 6 интерпретируется как команда остановки: "завершить работу программы". Заметим, что без команды JUMP (и даже без команды JZERO) можно обойтись ценой незначительного усложнения вычисления (одну команду можно проинтерпретировать несколькими другими). Например, вместо JZERO $m$ (в двух последовательных регистрах) можно записать немного более длинную последовательность в 12 регистрах $r_i, \dots, r_{i+11}$:

STORE 2
LOAD $i+2$
SUBT 2
STORE $i+11$
LOAD $n$
STORE 0

Здесь мы считаем, что регистр номер 2 свободен для рабочих целей, а регистр номер $n$ содержит необходимый адрес перехода. Первая команда записывает содержимое нулевого регистра в рабочий регистр. Вторая команда помещает единицу (код операции LOAD в регистре номер $i+2$) в нулевой регистр. В результате вычитания третьей командой из этой единицы старого содержимого нулевого регистра получается 1, если там был 0, и 0, если там было положительное число. Четвертая команда модифицирует адрес операнда шестой команды, устанавливая туда 1, если нужно произвести переход, и 0, если переход не нужен. Пятая команда помещает адрес перехода в нулевой регистр. Шестая (модифицированная) команда перемещает этот адрес в регистр 1 (регистр адреса выполняемой команды), если нужен переход (при этом считается, что обычное автоматическое увеличение первого регистра на 2 производится в самом начале выполнения каждой команды). Таким образом, если переход был необходим, то он выполнится, иначе будет выполняться следующая команда. Легко устранимый недостаток указанной последовательности команд – порча начального содержимого нулевого регистра.

Через $\mathrm{BFRAM}(M, K, P, N, T)$ будем обозначать множество всех свойств конечных последовательностей из $M$ натуральных чисел в диапазоне от 0 до $K-1$, распознаваемых равнодоступными адресными машинами размера $N$ с программами длины $P$ за время $T$. Для того, чтобы $\mathrm{BFRAM}(M, K, P, N, T)$ имело смысл, естественно, требуется выполнение условий: $P+M \leq N$ (возможность разместить программу и данные в адресуемой памяти) и $K \leq N$ (возможность разместить данные в ячейках). Очевидно, что $\mathrm{BFRAM}(M, K, P, N_1, T_1) \subseteq \mathrm{FRAM}(M, K, P, N_2, T_2)$ при $N_1 \leq N_2$ и $T_1 \leq T_2$.

## 2. Теорема об иерархии

Из сказанного выше следует, при всех натуральных $M$, $K$, $P$, $N$, $T$, $s_1$, $s_2$

$$\mathrm{BFRAM}(M, K, P, N, T) \subseteq \mathrm{BFRAM}(M, K, P, N+s_1, T+s_2).$$

Следующая теорема говорит о том, что при некоторых достаточно естественных предположениях данное включение является строгим.

**Теорема.** *При всех натуральных $M$, $N$, $P$ и при достаточно больших натуральных $c_1$, $c_2$ и $c_3$, если $c_1 \leq P$, $P+M \leq N$, $P \leq M$, $T < N$, то*

$$\mathrm{BFRAM}(M, N, P, N, T) \neq \mathrm{BFRAM}(M, N, P, N+c_1, c_2T+c_3P)$$

Данное утверждение выполняется для разных вариантов определения конкретной модели равнодоступной адресной машины рассматриваемого типа (в частности, оно в значительной мере не зависит от выбора системы команд и способа их кодировки). Этот результат говорит о том, что даже небольшое увеличение размера машины и времени ее работы приводит к возможности решения новых задач.

## 3. Доказательство теоремы об иерархии

Для доказательства теоремы достаточно построить программу $V$, удовлетворяющую ограничениям класса

$$\mathrm{BFRAM}(M, N, P, N + s_1, s_2T + s_3P)$$

при условиях теоремы для достаточно больших $s_1$, $s_2$ и $s_3$, и выдающую результат, отличный от всех результатов, выдаваемых программами $W$ с ограничениями класса

$$\mathrm{BFRAM}(M, N, P, N, T).$$

Заметим, что программа $V$ должна умещаться в отрезке памяти длины $P$. Требуемое построение производится диагональным методом: программа $V$ должна интерпретировать начальные $P$ ячеек своих входных данных как некоторую программу $W$ и моделировать $T$ шагов работы этой программы на данных, полученных из этой же самой $W$ дописыванием необходимого количества нулей в конец (напомним, что $P$ – длина $W$ – не больше $M$ – длины входных данных для $W$) и выдавать противоположный результат. Таким образом, программе $V$ класса

$$\mathrm{BFRAM}(M, N, P, N + s_1, s_2T + s_3P)$$

удастся отличиться поведением от всех программ класса

$$\mathrm{BFRAM}(M, N, P, N, T).$$

Приведем план построеня требуемой программы. Память моделирующей машины в начале работы распределяется следующим образом:

$$V, 0, \ldots, 0, W, 0, \ldots, 0, \ldots .$$

Здесь $V$ – программа (далее – нули, дополняющие длину до $P$), $W, 0, \ldots, 0$ – ее входные данные: первые $P$ ячеек – $W$, далее нули (рассматриваем поведение $W$ только в этом случае).

На первом этапе работы отрезок $W$ ($P$ ячеек, следующие за $V$) переписывается вплотную к $V$ и дублируется:

$$V', W, W, \ldots .$$

Здесь всесто $V$ записано $V'$ (той же длины), так как программа $V$ модифицируется в процессе работы.

На следующем этапе на $T$ шагов запускается интерпретатор RAM-$N$ с использованием отрезка памяти, содержащего в начале этого этапа

$$W, W, \ldots\ .$$

Таким образом, номера ячеек моделируемой машины равны номерам ячеек моделирующей машины, уменьшенным на длину программы $V$.

Интерпретатор пошагово (покомандно) моделирует работу машины. На каждом шаге сначала определяется, не является ли команда с адресом, заданным в моделируемом первом регистре командой остановки. Если это так, то работа завершается, иначе производится переход к одной из (четырех-шести, в зависимости от системы команд) подпрограмм интерпретатора, соответствующих выполняемым командам. Эта подпрограмма выполняет действия, описанные в определении RAM-$N$, с регистрами, имитируюшими регистры моделируемой машины. Попутно производится подсчет числа шагов и останов при исчерпании этого лимита. Размер интерпретатора невелик – порядка сотен регистров, так как количество выполняемых им действий сильно ограничено (например, четыре команды, каждая из которых изменяет не более двух регистров по простым правилам без проверки каких-либо условий).

В качестве примера приведем подпрограмму для интерпретации команды LOAD:

LOAD $P+1$
ADD $i+1$
STORE $i+7$
LOAD 0
ADD $i+15$
STORE $i+13$
LOAD 0
STORE $P$

Считаем, что подпрограмма расположена, начиная с регистра номер $i$. Первая команда выбирает содержимое моделируемого командного регистра адреса. Вторая преврашает его в реальный адрес следующего регистра (содержащего адрес операнда), прибавляя $P+1$. Третья команда модифицирует адрес операнда четвертой команды, заставляя ее выбирать сам адрес операнда.

Пятая команда вычисляет реальный адрес операнда по моделируемому, увеличивая его на $P$. Шестая команда модифицирует седьмую команду, заставляя ее выбирать сам операнд. Восьмая команда записывает выбранный операнд в регистр, моделирующий аккумулятор.

Некоторая трудность возникает при интерпретации команды сложения: для того, чтобы строящаяся программа всегда выдавала результат, необходимо избегать переполнений. Это легко сделать, сравнивая операнды с половиной $N$ и вычисляя расстояние от этих операндов, до половины $N$.

В результате работы интерпретатора не более, чем через $T$ шагов моделируемой машины в памяти окажется результат $q$ применения $W$ к $W$:

$$V'', q, \ldots,$$

где длины $V$ и $V''$ совпадают.

На заключительном этапе в ячейку $r_0$ записывается отрицание $q$ (1, если $q = 0$, иначе – 0).

Длину программы $V$ моделирующей машины обозначим через $s_1$. Заметим, что длина программы $V$ не зависит от величин $M$, $N$, $P$, $T$. Нетрудно показать, что моделирование можно производить с линейным замедлением: для моделирования $T$ шагов требуется $s_2T$ шагов моделирующей машины для некоторой константы $s_2$, не зависящей от $M$, $N$, $P$, $T$. Нетрудно также показать, что перемещение и копирование программы $W$ длины $P$ на первом этапе работы требует времени $s_3P$ для некоторой константы $s_3$, также не зависящей от $M$, $N$, $P$, $T$. Подводя итог, можно заключить, что для работы требуется $N + s_1$ ячеек памяти и время $s_2T + s_3P$ (время последнего этапа можно не учитывать, так как оно поглощается величиной $s_3P$ при $P$, больших $s_1$ по условию теоремы). Это завершает доказательство теоремы.

Автор не приводит точные значения констант $s_1$, $s_2$ и $s_3$, так как они зависят от конкретной системы команд и для рассматриваемой системы не представляют особого интереса. Нетрудно видеть, для этой и многих других естественных систем команд, аналогичных системам команд реальных компьютеров, порядки констант невелики: $s_1$ – от сотен до тысяч, $s_2$ – от десятков до сотен, $s_3$ – в пределах десятков.

## Заключение

Доказанная теорема показывает, что можно содержательно рассматривать классы машинной вычислительной сложности для функций с конечными областями определения, вычисляемых на конечных компьютерах. Эти классы образуют естественную "двумерную" иерархию по времени и объему памяти. Строгое вложение классов достигается при одновременном росте объема памяти и времени. При небольших временах (порядка общего объема памяти) для строгой иерархии достаточен линейный рост времени и рост памяти на постоянное число ячеек. Все это говорит о том, что для получения теорем о машинной вычислительной сложности бесконечно-асимптотический характер рассматриваемых утверждений не является необходимым.

## Литература

1. F. C. Hennie, R. E. Stearns, *Two-tape simulation of multitape Turing Machines.* JACM, **13**, No. 4 (1966), 533–546. Русск. перев. в кн.: "Проблемы математической логики" / Ред. В. А. Козмидиади, А. А. Мучник, М., "Мир", 1970, 194–212.
2. R. E. Stearns, J. Hartmanis, and P. M. Lewis II, *Hierarchies of memory limited computations.* IEEE Conf. Rec. Switch. Circuit Theory and Logic Design, N. Y., 1965, 179–190. Русск. перев. в кн.: "Проблемы математической логики" / Ред. В. А. Козмидиади, А. А. Мучник, М., "Мир", 1970, 301–319.

Beltiukov A. P. Combined hierarchies of finite random access machines.

A finitary approach to the computational complexity theory is considered in the article. Classes of *finite* random access machines are investigated. Combined space-and-time computational complexity hierarchies of predicates (properties, sets) recognizable by these machines are built. The corresponding theorem is proved.

Удмуртский университет
математический факультет
кафедра математического обеспечения ЭВМ
426034, Ижевск, ул. Университетская, 1
e-mail : belt@uni.udm.ru
http://ulm.uni.udm.ru/~belt

Поступило 29 апреля 2001 г.