ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Январь—март 2007. Серия 1. Том 14, № 1. 94–109

УДК 519.95

МОДЕЛИРОВАНИЕ НЕВЕТВЯЩИХСЯ ПРОГРАММ С УСЛОВНОЙ ОСТАНОВКОЙ НА УНИВЕРСАЛЬНОЙ МАШИНЕ ТЬЮРИНГА*)

А. В. Чашкин

Изучается время моделирования неветвящихся программ с условной остановкой на машине Тьюринга с тремя лентами, одна из которых используется для хранения программы, управляющей работой машины. Установлены неравенства, в которых среднее время моделирования оценивается через длину и среднее время работы моделируемых неветвящихся программ. Показано, что для некоторых программ полученные равенства являются точными с точностью до постоянного множителя.

Введение

Значительный интерес представляет задача сравнения вычислительной мощности различных моделей вычислений, в частности, вычислений, использующих условный переход, и вычислений без условного перехода. Типичными представителями вычислений первого типа являются различные варианты многоленточных машин Тьюринга. Основными моделями вычислений без условного перехода являются схемы (схемы из функциональных элементов, булевы схемы) и неветвящиеся программы. Описания и свойства этих моделей можно найти в [1, 3, 4]. Отметим, что несмотря на внешнее несходство определений схем и неветвящихся программ (в основе определения схемы лежит понятие графа, а неветвящиеся программы определяются как последовательности равенств), схемы и неветвящиеся программы являются практически одним и тем же объектом. Поэтому результаты о сложности схем почти дословно можно переносить на сложность неветвящихся программ и наоборот.

Оценки сложности схем, моделирующих работу машин Тьюринга, можно найти в [4], где, в частности, показано, что T шагов машины Тьюринга можно промоделировать схемой, сложность которой по порядку

*) Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований (проект 05–01–00994) и программы поддержки ведущих научных школ РФ (проект НШ–5400.2006.1).

не превосходит величины $T \log_2 T$. Связь длины программ универсальных машин Тьюринга и схемной сложности подробно рассматривается в [4, 6, 9], где под универсальной машиной Тьюринга понимается машина, способная моделировать при подходящей программе работу произвольной машины Тьюринга. В [9] установлено существование моделирующих программ, время моделирования которыми схем из L элементов есть $O(L \log_2 L)$.

В настоящей статье изучается время моделирования неветвящихся программ с условной остановкой (вычисляющих булевы функции) на универсальной трёхленточной машине Тьюринга. Работа организована следующим образом. В § 1 приводятся необходимые сведения о неветвящихся программах с условной остановкой и описываются преобразования, приводящие неветвящуюся программу к виду, удобному для моделирования на машине Тьюринга. В § 2 даётся описание универсальной машины Тьюринга и устанавливаются оценки времени выполнения некоторых элементарных преобразований. В § 3 формулируется и доказывается главный результат работы — теорема о времени моделирования неветвящихся программ на машине Тьюринга. В § 4 устанавливаются оценки среднего времени вычисления булевых функций на универсальной машине Тьюринга. Затем при помощи этих оценок устанавливается неулучшаемость по порядку главного результата работы.

Понятия, используемые ниже без определения, можно найти в [1, 3, 4].

§ 1. Неветвящиеся программы

Пусть $X = \{x_1, \ldots, x_n\}$ — множество независимых булевых переменных. Введём множество переменных $Y = \{y_1, \ldots, y_l\}$ и множество переменных $Z = \{z_1, \ldots, z_m\}$. Переменные из множества Y назовём *внутренними*, а переменные из множества Z — *выходными* переменными. Пусть, далее, $a \in Y \cup Z$, $b, c \in X \cup Y \cup Z$, f — булева функция, зависящая не более чем от двух переменных. *Вычислительной командой* p назовём выражение $p : a = f(b, c)$. Переменную a назовём *выходом* вычислительной команды p, а переменные b, c — *входами* этой команды. Если переменная a является выходом команды p, то будем говорить, что команда p изменяет значение этой переменной, а если a не является выходом p, то будем говорить, что команда p не изменяет её значение. *Командой остановки* p назовём выражение $p : \mathtt{Stop}(c)$. Переменную c назовём входом команды остановки p.

Последовательность $P = p_1 \ldots p_i \ldots p_L$, состоящая из вычислительных команд и команд остановки, называется *неветвящейся программой с условной остановкой*, если при любом $j \in \{1, 2, \ldots, L\}$ каждый вход команды p_j есть либо независимая переменная, либо выход некоторой вычислительной команды p_i, где $i < j$. Независимые переменные x_i будем называть входами программы P, а выходные переменные z_j — выходами этой программы.

Неветвящаяся программа работает в дискретные моменты времени $t = 0, 1, 2, \ldots$, не изменяет значения независимых переменных и изменяет значения внутренних и выходных переменных. Значения $y_i(x; t)$ внутренних переменных y_i и значения $z_j(x; t)$ выходных переменных z_j программы P в произвольный момент времени t на наборе независимых переменных $x = (x_1, \ldots, x_n)$ определим индуктивно.

1) В начальный момент времени $t = 0$ значения всех внутренних и выходных переменных считаем неопределёнными.

2) Если команда p_t не изменяет значения внутренней переменной y_i (или выходной переменной z_j), то положим $y_i(x; t) = y_i(x; t-1)$, $z_j(x; t) = z_j(x; t-1)$.

3) Если команда p_t изменяет значения внутренней переменной y_i (или выходной переменной z_j), и значения первого и второго входов команды p_t в момент времени $t-1$ равны соответственно $b(x; t-1)$ и $c(x; t-1)$, то положим

$$y_i(x; t) = f_t(b(x; t-1), c(x; t-1)), \quad z_j(x; t) = f_t(b(x; t-1), c(x; t-1)).$$

Значением команды p_t программы P на наборе σ значений независимых переменных $x = (x_1, ..., x_n)$ назовём значение её выхода в момент времени t и обозначим через $p_t(\sigma)$.

Через $n_P(p)$ обозначим номер команды p в программе P, т. е. $n_P(p_i)=i$. Пусть $p_{t_1}, \ldots, p_{t_r}$ — все команды остановки из P, причём $t_1 < \cdots < t_r$. Тогда через s_j будем обозначать j-ю команду остановки программы P, т. е. $s_j \equiv p_{t_j}$. Вычислительную команду p_i (переменную x_l) назовём *нулевым аргументом* команды остановки s_j, $n(s_j) = r$, и обозначим через q_j, если:

(i) выход команды p_i (переменная x_l) является входом команды s_j.

(ii) среди команд p_t, $i < t < r$, нет команды, выход которой совпадает с выходом команды p_i.

Будем говорить, что k-я команда остановки s_k прекращает вычисления программы P на наборе σ, если

$$q_1(\sigma) = \cdots = q_{k-1}(\sigma) = 0, \quad q_k(\sigma) = 1.$$

Результат действия программы P на наборе σ обозначим через $P(\sigma)$ и его l-ю компоненту $P_l(\sigma)$ определим следующим образом:

$$P_l(\sigma) = \begin{cases} z_l(\sigma; t_k), & \text{если } q_1(\sigma) = \dots = q_{k-1}(\sigma) = 0, \quad q_k(\sigma) = 1, \\ z_l(\sigma; L), & \text{если } q_1(\sigma) = \dots = q_r(\sigma) = 0, \end{cases}$$

т. е. $P_l(\sigma)$ равно значению l-й выходной переменной z_l в момент остановки программы. Легко видеть, что

$$\begin{aligned} P_l(x) = q_1(x) z_l(x; t_1) \vee \overline{q}_1(x) q_2(x) z_l(x; t_2) \vee \dots \\ \vee \overline{q}_1(x)\overline{q}_2(x)\dots\overline{q}_{k-1}(x) q_k(x) z_l(x; t_k) \vee \dots \\ \vee \overline{q}_1(x)\overline{q}_2(x)\dots\overline{q}_{r-1}(x) q_r(x) z_l(x; t_r) \vee \overline{q}_1(x)\overline{q}_2(x)\dots\overline{q}_r(x) z_l(x; L). \end{aligned} \quad (1)$$

Сложностью $C(P)$ программы P назовём число команд этой программы. *Временем работы* $T_P(\sigma)$ программы P на наборе σ значений переменных x назовём минимальное $n(s_j)$ такое, что $q_j(\sigma) = 1$, т. е. это число команд, выполненных до остановки программы. Если все $q_j(\sigma) = 0$, то выполняются все команды программы и $T_P(x) = C(P)$. Величину

$$T(P) = 2^{-n} \sum T_P(\sigma),$$

где суммирование производится по всем двоичным наборам σ длины n, назовём *средним временем работы* программы P. Если для некоторой системы булевых функций (булевой вектор-функции) f и любого двоичного набора σ справедливо равенство $f(\sigma) = P(\sigma)$, то будем говорить, что программа P вычисляет вектор-функцию f. Величину

$$T(f) = \min T(P),$$

где минимум берётся по всем программам, вычисляющим f, назовём *средним временем вычисления* (*средней сложностью*) вектор-функции f. Программу P, вычисляющую такую f, что $T(P) = T(f)$, назовём *минимальной* программой. Величину $C(f) = \min C(P)$, где минимум берётся по всем программам, вычисляющим f, назовём *сложностью* вектор-функции f. Величина $C(f)$ характеризует время, необходимое для вычисления f в худшем случае. Поэтому $C(f)$ также будем называть сложностью в худшем случае.

Лемма 1. *Произвольная программа P с n входами и m выходами может быть преобразована в такую программу P' без команд остановки, что $P'(x) = P(x)$ для любого x и $C(P') \leqslant 4C(P) - 4$.*

Доказательство. Пусть $P = p_1 \ldots p_L$ — произвольная неветвящаяся программа, $s_1, \ldots, s_r$ — все её команды остановки, $q_1, \ldots, q_r$ — нулевые аргументы команд остановки. Как и в (1) полагаем, что i-я команда остановки s_i программы P является её t_i-й командой. Введём вспомогательные функции h_i и h'_i. Положим

$$h'_0(x) \equiv 1, \qquad h'_k(x) = \bigwedge_{i=1}^{k} \overline{q}_i(x) \qquad \text{при } k \in \{1, 2, \ldots, r\},$$
$$h_{r+1}(x) = h'_r(x), \quad h_k(x) = h'_{k-1}(x) q_k(x) \text{ при } k \in \{1, 2, \ldots, r\}.$$

Используя введённые функции h_i, преобразуем (1):

$$P_l(x) = h_1(x) z_l(x; t_1) \vee \ldots \vee h_r(x) z_l(x; t_r) \vee h_{r+1}(x) z_l(x; L). \tag{2}$$

Теперь при всех i, j таких, что $1 \leqslant i < j \leqslant r + 1$, определим функции

$$h_{i,j}(x) = \bigvee_{k=i}^{j} h_k(x).$$

Предположим, что l-я выходная переменная программы P вычисляется только перед первой командой остановки и после команд остановки, индексы которых принадлежат множеству $\{i_1, \ldots, i_k\}$, т. е.

$$z_l(x; t_i) = \begin{cases} z_l(x; t_1) & \text{при всех } i \in \{1, \ldots, i_1\} \\ z_l(x; t_{i_s+1}) & \text{при всех } i \in \{i_s + 1, \ldots, i_{s+1}\}. \end{cases}$$

Тогда равенство (2) после несложных преобразований приводится к виду

$$\begin{aligned} P_l(x) = h_{1,i_1}(x) z_l(x; t_1) \vee h_{i_1+1,i_2}(x) z_l(x; t_{i_1+1}) \vee \ldots \\ \vee h_{i_{k-1}+1,i_k}(x) z_l(x; t_{i_{k-1}+1}) \vee h_{i_k+1,r+1}(x) z_l(x; t_{i_k+1}). \end{aligned} \tag{3}$$

Легко видеть, что функции h_i и h'_j определены так, что

$$h_i(x) h'_j(x) = \begin{cases} 0 & \text{при } i \leqslant j, \\ h_i(x) & \text{при } i > j. \end{cases}$$

Поэтому $h_{1,j}(x) h'_i(x) = h_{i+1}(x) \vee \cdots \vee h_j(x) = h_{i+1,j}(x)$ при $i < j$, т. е. при вычисленных функциях $h_{1,i}$ и h'_i для вычисления каждой функции $h_{i,j}$, встречающейся в (3), достаточно одной вычислительной команды.

Преобразуем программу P в программу P', которая состоит только из вычислительных команд и вычисляет ту же булеву функцию, что и P.

Число вычислительных команд программы P обозначим через L_1. Тогда $L = L_1 + r$. Преобразование программы P состоит в следующем:

1) Вычисляются все функции h_i, h'_i и $h_{1,i}$. Для этого потребуется $3r-2$ новых команд.

2) Вычисляются все необходимые функции $h_{i,j}$. Для этого потребуется столько новых команд, сколько раз в программе P вычисляются выходные переменные. Так как каждый раз каждая выходная переменная вычисляется собственной вычислительной командой, то потребуется не более L_1 команд.

3) В соответствии с равенством (3) вычисляются все компоненты P_l. Для этого потребуется не более $2L_1 - m \leqslant 2L_1 - 1$ новых команд.

4) Удаляются все команды остановки.

Легко видеть, что общее число новых команд не превосходит $3L - 3$. Следовательно, сложность программы P' не превосходит $4L-4$. Лемма 1 доказана.

Для произвольной программы P определим такую программу $P^\star$, что

$$P^\star(x) = P(x),\ T_{P^\star}(x) \leqslant 8T_P(x) \tag{4}$$

для любого x. Далее программу $P^\star$ будем называть $\star$-программой.

Пусть P — такая программа с n входами и m выходами, что $C(P)=L$. Программу P преобразуем в программу P', введя в P новые команды остановки $s'_1, \ldots, s'_r$ так, чтобы номера этих команд в программе P' были последовательными степенями двойки, причём $n_{P'}(s'_1) = 2^{\lceil \log_2 n_P(s_1) \rceil}$, а нулевые аргументы — нулевыми аргументами непосредственно предшествующих им команд остановки программы P. Очевидно, что $P(x) = P'(x)$ для любого x. Программу P' представим в виде последовательности подпрограмм P_i, разделённых командами s'_i, т. е.

$$P' = P_1 s'_1 P_2 s'_2 \ldots P_i s'_i P_{i+1} \ldots P_r s'_r P_{r+1}.$$

Каждую подпрограмму P_i преобразуем в такую программу P'_i без команд остановки, как в лемме 1. В результате этого преобразования получим новую программу $P^\star$. Без ограничения общности будем считать, что номера команд s'_i в программе $P^\star$ по-прежнему являются степенями двойки*). Из леммы 1 следует, что $C(P^\star) \leqslant 4C(P)$ и программа $P^\star$ удовлетворяет соотношениям (4).

*) Выполнения этого условия можно добиться при помощи введения дополнительных команд, не влияющих на результат работы программы.

§ 2. Машины Тьюринга

Машиной Тьюринга **M** будем называть конечный автомат Q, снабжённый тремя двухсторонними лентами A, B и C, ячейки которых пронумерованы целыми числами так, что ячейки с меньшими номерами находятся левее ячеек с бóльшими номерами. В каждый момент времени автомат имеет доступ к одной (активной) ячейке каждой ленты и в зависимости от величин, содержащихся в активных ячейках лент, и своего внутреннего состояния может:

изменить содержимое активных ячеек на лентах A и B;

сдвинуть каждую из лент на одну ячейку либо вправо, либо влево;

изменить внутреннее состояние автомата Q.

Указанные действия составляют один такт работы машины.

Ленты A и B называются рабочими, используются для записи промежуточных результатов вычислений и в каждый момент времени либо пусты, либо содержат только булевы величины. Лента C называется программной. Её алфавит состоит из конечного числа элементов $c_1, \dots, c_k$, которые будем называть командами. Последовательность команд, находящаяся на ленте C, называется программой, управляющей работой машины. Будем полагать, что число команд и число внутренних состояний автомата Q достаточно велики для того, чтобы машина **M** могла выполнять описываемые ниже действия.

Будем говорить, что машина **M** под управлением программы R вычисляет систему булевых функций $f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)$, если:

(i) В начальный момент времени значения переменных $x_1, \dots, x_n$ расположены последовательно (в соседних ячейках) на ленте A. Команды программы R расположены на ленте C. На лентах A и C активными являются самые левые непустые ячейки. Лента B пуста.

(ii) После окончания работы машины значения $f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)$ последовательно расположены на ленте A. На ленте A активной является самая левая непустая ячейка. Лента B пуста.

Результат работы машины **M** под управлением программы R (или короче результат работы программы R) над набором переменных x будем обозначать через $R(x)$.

Сложностью $C_{\mathbf{M}}(R)$ программы R назовём число команд этой программы. Пусть машина **M** под управлением программы R вычисляет вектор-функцию $f(x)$. *Временем работы* $T_{\mathbf{M}R}(\sigma)$ машины **M** под управлением программы R на наборе σ значений переменных x назовём число тактов работы, выполненных до остановки машины. Величину $T_{\mathbf{M}}(R) =$

$2^{-n}\sum T_{\mathbf{M}R}(\sigma)$, где суммирование производится по всем двоичным наборам σ длины n, назовём *средним временем работы* программы R. Величину $T_{\mathbf{M}}(f) = \min T_{\mathbf{M}}(R)$, где минимум берётся по всем программам, вычисляющим f, назовём *средним временем вычисления* вектор-функции f на машине $\mathbf{M}$. Программу R такую, что $T_{\mathbf{M}}(R) = T_{\mathbf{M}}(f)$, будем называть *минимальной программой* вектор-функции f.

Пусть неветвящаяся программа P вычисляет систему булевых функций $f_1(x_1,\ldots,x_n),\ldots,f_m(x_1,\ldots,x_n)$. Будем говорить, что машина $\mathbf{M}$ под управлением программы R моделирует работу P, если она вычисляет эту систему функций.

Рассмотрим список $l = (l_1,\ldots,l_n)$ и двоичный набор $\alpha = (\alpha_1,\ldots,\alpha_n)$. Из элементов списка l составим два новых списка l_α и ${}_\alpha l$. Первый список сформируем следующим образом. Сначала в порядке возрастания индексов выпишем все элементы l_i списка l, для которых $\alpha_i = 0$. Затем также в порядке возрастания индексов выпишем все оставшиеся элементы. Например, если $l = (12345)$ и $\alpha = (10011)$, то $l_\alpha = (23145)$. Список l_α назовём α-перестановкой списка l. Определим второй список. Допустим, что $\sum\limits_{i=1}^{n}\alpha_i = k$. Тогда список ${}_\alpha l$ получим из набора α, заменив его i-й нуль $(1 \leqslant i \leqslant n-k)$ элементом l_i, а j-ю единицу $(1 \leqslant j \leqslant k)$ элементом l_{n-k+j}. Например, если $l = (12345)$ и $\alpha = (10011)$, то $l_\alpha = (31245)$. Список ${}_\alpha l$ назовём обратной α-перестановкой списка l.

Будем говорить, что машина Тьюринга $\mathbf{M}$ выполняет α-перестановку (обратную α-перестановку) n-элементного списка l, если: перед началом работы элементы списка l последовательно расположены на ленте А и самая левая непустая ячейка ленты А является активной; во время работы машины каждая ячейка, расположенная на ленте А, правее ячеек с элементами списка, не является активной; после окончания работы элементы списка l_α (списка ${}_\alpha l$) последовательно расположены на ленте А и самая левая непустая ячейка ленты А является активной. Нетрудно видеть, что выполнение обратной α-перестановки позволяет слить два упорядоченных списка в один упорядоченный список.

Лемма 2. (i) *Произвольную α-перестановку n-элементного списка l можно выполнить на машине Тьюринга* $\mathbf{M}$ *за время, не превосходящее* $4n$;

(ii) *произвольную обратную α-перестановку n-элементного списка l можно выполнить на машине Тьюринга* $\mathbf{M}$ *за время, не превосходящее* $4n$.

Доказательство. Оба утверждения леммы доказываются аналогично, поэтому ограничимся доказательством только первого утверждения.

Без ограничения общности будем полагать, что элементы списка находятся на ленте A в ячейках с номерами от 1 до n, а на ленте B активной является ячейка с номером 1. Последовательно перенесём все элементы списка l_i, для которых $\alpha_i = 0$, с ленты A на ленту B. Для этого потребуется не более n шагов. Затем сдвинем ленту A вправо так, чтобы на ней активной стала первая ячейка. Для этого также потребуется не более n шагов. После этого перенесём на ленту B оставшиеся на ленте A элементы списка l_i (для каждого из этих элементов $\alpha_i = 1$). На это потребуется не более n шагов. Наконец, перемещая ленты A и B вправо, за n шагов перенесём элементы списка с ленты B на ленту A. Поэтому общее время не превосходит $4n$. Лемма 2 доказана.

Без доказательства приведём простое следствие леммы 2.

Лемма 3. *Перестановку n-элементного списка можно выполнить на машине Тьюринга $\mathbf{M}$ за время, не превосходящее $8n\log_2 n$.*

§ 3. Основная теорема

Основным результатом является является следующее утверждение.

Теорема 1. *Пусть P — произвольная неветвящаяся программа с n входами и m выходами. Тогда для машины Тьюринга $\mathbf{M}$ найдётся такая программа R, что машина $\mathbf{M}$ под управлением этой программы моделирует работу программы P так, что для каждого x*

$$T_{\mathbf{M}R}(x) \leqslant c_1 n\log_2 n + c_2 T_P(x)\log_2 T_P(x),$$

где c_1 и c_2 — константы.

Прежде чем доказывать теорему 1, установим важный вспомогательный результат [9], относящийся к моделированию программ без команд остановки.

Лемма 4. *Пусть P — произвольная неветвящаяся программа без команд остановки с n входами и m выходами. Тогда для машины Тьюринга $\mathbf{M}$ найдется такая программа R, что машина $\mathbf{M}$ под управлением этой программы моделирует работу программы P так, что время моделирования программы P не превосходит величины $c_3 C(P)\log_2 C(P)$, где c_3 — константа.*

Доказательство. Нетрудно видеть, что моделирование программы, состоящей из одной команды, потребует не более четырёх шагов. В качестве примера рассмотрим неветвящуюся программу P, которая вычисляет трёхкомпонентную вектор-функцию $(x, y, f(x, y))$. Допустим, что её аргументы x и y перед началом работы находятся на ленте A в двух соседних ячейках, из которых левая является активной ячейкой этой ленты. Тогда моделирование программы P можно провести следующим образом. На первом шаге автомат Q запоминает символ x из первой ячейки ленты A и лента A сдвигается влево. На втором шаге автомат Q запоминает символ y из второй ячейки ленты A, записывает в активную ячейку ленты A значение $f(x, y)$ и лента A сдвигается вправо. На третьем шаге автомат Q записывает в активную ячейку ленты A символ y и лента A сдвигается вправо. На четвёртом шаге в активную ячейку ленты A помещается символ x. В результате в трёх соседних ячейках ленты A будут записаны значения вектор-функции $(x, y, f(x, y))$, а самая левая непустая ячейка ленты A окажется активной.

Теперь допустим, что любая неветвящаяся программа, состоящая из 2^t команд, где $1 \leqslant t < k$, может быть промоделирована на машине Тьюринга так, что для времени $T(2^t)$ моделирования справедливо неравенство $T(2^t) \leqslant 14(t+1)2^t$; при этом будут выполнены следующие условия:

(i) В начальный момент времени входные данные (не более $2 \cdot 2^t$ символов) последовательно расположены на ленте A в произвольном порядке. На ленте A активной является самая левая непустая ячейка. Лента B пуста.

(ii) Во время работы машины ни одна из ячеек, расположенных на ленте A правее ячеек с входными данными, не является активной.

(iii) После окончания работы машины результаты вычислений, не более $3 \cdot 2^t$ символов (аргументы программы и вычисленные значения), последовательно расположены на ленте A в произвольном порядке. На ленте A активной является самая левая непустая ячейка. Лента B пуста.

Основываясь на сделанном предположении, опишем конструкцию программы R_k, которая моделирует неветвящуюся программу из 2^k команд с $p \leqslant 2 \cdot 2^k$ входами и $q \leqslant 3 \cdot 2^k$ выходами. Команды программы R_k разобьём на следующие четыре последовательных блока:

1) Команды первого блока переупорядочивают данные на ленте A так, чтобы данные, используемые первыми 2^{k-1} командами неветвящейся программы, были расположены левее данных, не используемых этими командами. В соответствии с леммой 2 для перемещения аргументов первых 2^{k-1} команд неветвящейся программы потребуется не более $4 \cdot 2^k$ шагов.

2) Команды второго блока выполняют вычисления, соответствующие первым 2^{k-1} командам неветвящейся программы, и удовлетворяют усло-

виям (i)–(iii). В соответствии с предположением индукции для этого потребуется не более $T(2^{k-1})$ шагов.

3) Команды третьего блока переупорядочивают данные на ленте А так, чтобы данные, используемые последними 2^{k-1} командами неветвящейся программы, были расположены левее данных, не используемых этими командами. После выполнения команд первого блока на ленте А может оказаться не более $3\cdot 2^{k-1}$ значений, не являющихся аргументами последних 2^{k-1} команд неветвящейся программы. В соответствии с леммой 2 для перемещения аргументов последних 2^{k-1} команд неветвящейся программы потребуется не более $4\bigl(3\cdot 2^{k-1}+2\cdot 2^{k-1}\bigr)$ шагов.

4) Команды четвертого блока выполняют вычисления, соответствующие последним 2^{k-1} командами неветвящейся программы, и удовлетворяют условиям (i)–(iii). В соответствии с предположением индукции для этого потребуется не более $T(2^{k-1})$ шагов.

Индукцией по k покажем, что время $T(2^k)$ работы программы R_k удовлетворяет неравенству $T(2^k)\leqslant 14(k+1)2^k$. При $k=0$ это неравенство очевидно, так как моделирование неветвящейся программы, состоящей из одной команды, выполняется не более чем за четыре шага. Допустим, что $T(2^{k-1})\leqslant 14k2^{k-1}$ при $k\geq 2$. Тогда из пп. 1)–4) имеем

$$T(2^k)\leqslant 2T(2^{k-1})+28\cdot 2^{k-1}\leqslant 2\cdot 14k2^{k-1}+28\cdot 2^{k-1}=14(k+1)2^k.$$

Теперь завершения моделирования программы P достаточно упорядочить данные на ленте А. Из леммы 3 следует, что это можно сделать за время, не превосходящее $8\cdot 5\cdot 2^k(k+3)$. Лемма 4 доказана.

Доказательство теоремы 1. Преобразуем программу P в программу $P^\star$, удовлетворяющую соотношениям (4). Далее будем моделировать работу программы $P^\star$. Сначала будем полагать, что перед началом работы машины $\mathbf{M}$ значения независимых переменных на ленте А расположены в таком порядке, в каком они используются при вычислениях в программе $P^\star$, т. е. чем раньше переменная x_i встречается в виде аргумента какой-либо команды в $P^\star$, тем левее на ленте А находится её значение. Пусть $P^\star=P_1s_1P_2s_2\dots P_is_iP_{i+1}\dots P_rs_rP_{r+1}$, где s_i — команды остановки, P_i — подпрограммы без команд остановки. Программу R, моделирующую программу $P^\star$, представим в виде последовательности блоков $R_1,\dots,R_{r+1}$. Первые r блоков устроены одинаково и каждый блок R_i состоит из двух частей R_i^1 и R_i^2. В каждом таком блоке первая часть R_i^1 моделирует подпрограмму P_i так, как это было сделано в доказательстве леммы 4, причём после окончания работы R_i^1 на ленте A в самых левых m непустых ячейках в порядке возрастания значений индекса j расположены значения z_j, а в следующей правой ячейке находится значение нулевого аргумента q_i команды остановки s_i и эта ячейка является активной. Часть R_i^2 состоит из единственной команды, результатом действия которой является стирание содержимого всех ячеек, находящихся правее ячеек со значениями z_j при $q_i = 1$, а при $q_i = 0$ сдвиг ленты A вправо так, что активной ячейкой становится самая левая непустая ячейка ленты A. Последний блок R_{r+1} моделирует подпрограмму P_{r+1}.

Если работу программы $P^\star$ на наборе x останавливает команда s_j, то из леммы 4 следует, что

$$
\begin{aligned}
T_{\mathbf{M}R}(x) \leqslant \sum_{i=1}^{k} c_3 n_{P^\star}(s_i)\log_2 n_{P^\star}(s_i)& \\
\leqslant c_4 n_{P^\star}(s_j)\log_2 n_{P^\star}(s_j) &= c_4 T_{P^\star}(x)\log_2 T_{P^\star}(x),
\end{aligned}
$$

где c_4 — некоторая константа.

Теперь для завершения доказательства теоремы осталось воспользоваться неравенством из (4) и избавиться от условия предварительной упорядоченности значений независимых переменных на ленте A до начала работы. Это можно легко сделать при помощи леммы 3, если до начала работы программы R переставить за время $O(n\log_2 n)$ значения независимых переменных так, как это требуется для работы R. Теорема 1 доказана.

Пусть программа R моделирует работу неветвящейся программы P так, что для любого x время $T_{\mathbf{M}R}(x) = \Theta(T_P(x))\log_2 T_P(x))$. Оценим среднее время работы $T_{\mathbf{M}}(R)$ программы R в терминах сложности и средней сложности программы P. Из выпуклости вниз функции $t\log_2 t$ следует, что

$$
\begin{aligned}
T_{\mathbf{M}}(R) = 2^{-n}\sum_x T_{\mathbf{M}R}(x) \geqslant 2^{-n}\sum_x c_5 T_P(x)\log_2 T_P(x)& \\
\geqslant c_5\Big(2^{-n}\sum_x T_P(x)\Big)\log_2\Big(2^{-n}\sum_x T_P(x)\Big) = c_5 T(P)\log_2 T(P),& \quad (5)
\end{aligned}
$$

где c_5 — некоторая константа. С другой стороны, так как $T_P(x) \leqslant C(P)$ для любого x, то

$$
\begin{aligned}
T_{\mathbf{M}}(R) = 2^{-n}\sum_x T_{\mathbf{M}R}(x) \leqslant 2^{-n}\sum_x c_6 T_P(x)\log_2 T_P(x)& \\
\leqslant 2^{-n}\sum_x c_6 T_P(x)\log_2 C(P) = c_6 T(P)\log_2 C(P),& \quad (6)
\end{aligned}
$$

где c_6 — константа.

Из (6), (5) и теоремы 1 следует, что, если для программы P с n входами $T(P) \gg n$, то существует такая моделирующая P программа R, что

$$c_5 T(P) \log_2 T(P) \leqslant T_{\mathbf{M}}(R) \leqslant c_6 T(P) \log_2 C(P). \tag{7}$$

§ 4. Булевы функции

В этом разделе покажем, что неравенство из теоремы 1 является точным по порядку. Для этого установим следующий результат.

Теорема 2. *Пусть $n \log_2 n \ll k \leqslant 2^{n-1}$, $P_2(k,n)$ — множество булевых функций от n переменных, каждая из которых равна единице ровно на k наборах. Тогда найдутся такие константы c_7 и c_8, что $T_{\mathbf{M}}(f) \geqslant c_7 k$ для почти каждой булевой функции f из $P_2(k,n)$ при $n \to \infty$ и $T_{\mathbf{M}}(f) \leqslant c_8 k$ для каждой булевой функции f из $P_2(k,n)$.*

Доказательство. Пусть f — булева функция от n переменных, R — программа, под управлением которой $\mathbf{M}$ вычисляет f. Каждому двоичному набору x длины n, рассматриваемому как двоичная запись натурального числа, поставим в соответствие его номер $N_R(x)$ такой, что $1 \leqslant N_R(x) \leqslant 2^n$; $N_R(x) < N_R(y)$, если $T_{\mathbf{M}R}(x) < T_{\mathbf{M}R}(y)$; $N_R(x) < N_R(y)$, если $T_{\mathbf{M}R}(x) = T_{\mathbf{M}R}(y)$ и $x < y$.

Оценим сверху число булевых функций из $P_2(k,n)$, каждая из которых может быть вычислена на машине Тьюринга со средним временем, не превосходящим величины $c_7 k$. Пусть f — одна из таких функций, R — минимальная программа, вычисляющая f. Рассмотрим набор x_0 такой, что $N_R(x_0) = 2^{n-1}$. Тогда из определения средней сложности следует, что

$$T_{\mathbf{M}}(R) = 2^{-n} \sum_{y} T_{\mathbf{M}R}(y) > 2^{-n} \sum_{y \,\mid\, N(y) > N(x_0)} T_{\mathbf{M}R}(y) \geqslant \frac{1}{2} T_{\mathbf{M}R}(x_0).$$

Поэтому $T_{\mathbf{M}R}(x_0) < 2T_{\mathbf{M}}(f)$. Так как $T_{\mathbf{M}}(f) \leqslant c_7 k$, то легко видеть, что

$$T_{\mathbf{M}R}(x_0) < 2c_7 k.$$

Каждая функция однозначно определяется первыми $T_{\mathbf{M}R}(x_0)$ командами своей программы R и двоичным вектором длины не более 2^{n-1}, состоящим из значений функции f на тех аргументах, время работы R на которых больше времени работы этой программы на x_0. Пусть K — число различных команд, входящих в систему команд машины $\mathbf{M}$. Тогда число функций f таких, что средняя сложность $T_{\mathbf{M}}(f)$ не превосходит c_7k, ограничена сверху величиной

$$K^{c_7k}\sum_{i=0}^{2^{n-1}}\binom{2^{n-1}}{i}\leqslant K^{c_7k}k\binom{2^{n-1}}{k}\leqslant 2^{c_7k\log_2 K-k+\log_2 k}\binom{2^n}{k},$$

которая при $c_7\log_2 K\leqslant\frac{1}{2}$ и $n\to\infty$ есть $o\left(\binom{2^n}{k}\right)$. Таким образом, средняя сложность почти каждой булевой функции из $P_2(k,n)$ не меньше c_7k. Первое неравенство теоремы доказано.

Верхняя оценка является тривиальным следствием теоремы 1 и главного результата из [8]: $T(f)=O(\frac{k}{\log_2 k})$ для любой функции f из $P_2(k,n)$ при условии, что $n\log_2 n\ll k\leqslant 2^{n-1}$. Теорема 2 доказана.

Теперь для того чтобы показать, что неравенство из теоремы 1 является точным по порядку, достаточно рассмотреть такую функцию f из $P_2(k,n)$, где $n\log_2 n\ll k\leqslant 2^{n-1}$, что $T(f)=\Theta\left(\frac{k}{\log_2 k}\right)$. Из [5, 8, 3] следует, что для такой функции $\log_2 T(f)=\Theta(\log_2 C(f))=\Theta(\log_2 k)$. Пусть P — минимальная неветвящаяся программа функции f. Если неравенство из теоремы 1 не является точным по порядку, то согласно (7) найдётся такая программа R, что $T_{\mathbf{M}}(R)=o(T(P)\log_2 T(P))$. Но тогда $T_{\mathbf{M}}(f)=o(k)$, что противоречит первому неравенству из теоремы 2.

Наконец покажем, что в неравенствах (7) могут достигаться как верхняя так и нижняя оценки. Пусть $k=\lceil(n+\log n)/2\rceil$ и g — любая из «почти всех» самых сложных булевых функций от k переменных, т. е. $C(g)=\Theta(\sqrt{2^n/n}$. Рассмотрим функцию

$$f(x_1,\dots,x_n)=\overline{x}_1\&\cdots\&\overline{x}_{n-k}\&g(x_{n-k+1},\dots,x_n).$$

Очевидно, что $C(f)=\Theta(\sqrt{2^n/n})$, и, кроме того, нетрудно показать (подробности см. в [5, 7]), что $T(f)=O(1)$. Пусть R — произвольная программа, которая вычисляет f, R' — такая программа, которая вычисляет g и $T_{\mathbf{M}}(R')=T_{\mathbf{M}}(g)$. Из теоремы 2 следует, что $T_{\mathbf{M}}(g)\geqslant c_7 2^k$. Поэтому

$$\begin{aligned}T_{\mathbf{M}}(f)&=2^{-n}\sum_x T_{\mathbf{M}R}(x)\geqslant 2^{-n}\sum_{x_1=\dots=x_{n-k}=0}T_{\mathbf{M}R}(x)\\&\geqslant 2^{-n+k}\Big(2^{-k}\sum_{x_{n-k+1},\dots,x_n}T_{\mathbf{M}R'}(x)\Big)\geqslant 2^{-n+k}T_{\mathbf{M}}(R')\geqslant 2^{-n+2k}c_7=\Theta(n).\end{aligned}$$

Следовательно, если P — минимальная неветвящаяся программа функции f, R — программа, моделирующая P так, как это было описано в доказательстве леммы 4, то

$$T_{\mathbf{M}}(R) = \Theta(T(P)\log_2 C(P)) = \Theta(n), \quad T(P)\log_2 T(P) = O(1),$$

т. е. в (7) достигается верхняя граница.

Пусть $k = \lceil n/2 \rceil$, а g — снова любая из «почти всех» самых сложных булевых функций от k переменных. В этом случае $C(g) = \Theta(\sqrt{2^n}/n)$. Рассмотрим функцию $f(x_1, \dots, x_n) = \overline{x}_1 \& \cdots \& \overline{x}_{n-k} \& g(x_{n-k+1}, \dots, x_n)$ и вычисляющую её неветвящуюся программу P, в которой после первых $n-k+1$ команд

$$z = 0,\ \mathtt{Stop}(x_1),\ \mathtt{Stop}(x_2), \dots, \mathtt{Stop}(x_{n-k})$$

находятся команды минимальной программы функции g. Как и в предыдущем случае нетрудно показать, что $C(P) = \Theta(\sqrt{2^n}/n)$ и $T(P) = O(1)$. Если R — программа, моделирующая P так, как это было описано в доказательстве леммы 4, то

$$T_{\mathbf{M}}(R) = O\Big(2^{-n}\Big(\sum_{j=1}^{n-k} 2^{n-j} j \log_2 j + 2^k \frac{2^k}{k} \log_2 \frac{2^k}{k}\Big)\Big) = O(1).$$

Следовательно,

$$T_{\mathbf{M}}(R) = \Theta(T(P)\log_2 T(P)) = O(1), \qquad T(P)\log_2 T(P) = \Theta(n),$$

т. е. в (7) достигается нижняя граница.

ЛИТЕРАТУРА

1. **Ахо А., Хопкрофт Дж., Ульман Дж.** Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.

2. **Лупанов О. Б.** Об одном подходе к синтезу управляющих систем — принципе локального кодирования // Проблемы кибернетики. Вып 14. М.: Наука, 1965. С. 31–110.

3. **Мальцев А. И.** Алгоритмы и рекурсивные функции. М.: Наука, 1986.

4. **Сэвидж Д. Э.** Сложность вычислений. М., 1998.

5. **Чашкин А. В.** О среднем времени вычисления значений булевых функций // Дискрет. анализ и исслед. операций. Сер. 1. 1997. Т. 4, № 1. С. 60–78.

6. **Чашкин А. В.** Моделирование схем из функциональных элементов машинами Тьюринга // Дискрет. анализ и исслед. операций. Сер. 1. 1999. Т. 5, № 3. С. 42–70.

7. **Чашкин А. В.** Средняя сложность булевых функций // Дискретная математика и её приложения. Сборник лекций. Вып 1. М.: Изд-во механико-математического факультета МГУ, 2001. С. 145–170.

8. **Чашкин А. В.** Об одном методе вычисления частичных булевых функций // Математические вопросы кибернетики. Вып 12. М.: Физматлит, 2003. С. 231–246.

9. **Чашкин А. В.** Моделирование схем из функциональных элементов на универсальной машинаме Тьюринга // Дискретная математика. 2004. Т. 16, вып. 2. С. 98–103.

Адрес автора:

МГУ, мех.-мат. факультет,
Воробьёвы горы,
119992 Москва, Россия.
E-mail: chash@online.ru

Статья поступила
26 октября 2006 г.