ОБЩАЯ ТЕОРИЯ АЛГОРИТМОВ

Глава 5, стр. 114

- Необходимость нумерации произвольных объектов вызвана, прежде всего, необходимостью анализа различных задач, которые должны обрабатывать алгоритмы в качестве исходной информации.
- Следовательно, для того, чтобы рассматривать алгоритмы над алгоритмами, необходимо представлять алгоритм (в данной главе — машину Тьюринга) в виде натуральных чисел.

Геделевская нумерация объектов

Считается, что введена ***система Геделевской нумерации*** для всех объектов *A*, принадлежащих некоторому множеству *M*, если выполняются следующие два требования:

1. существует натуральное число *ng*(*A*), которое однозначно определяется по *A*;
2. для всех *n*, принадлежащих множеству натуральных чисел *N*, выполняется одно из двух условий:
   - либо не существует объекта *A*, принадлежащего множеству *M*, такого, что *n* = *ng*(*A*);
   - либо существует единственный объект *A*, принадлежащий *M*, такой, что *n* =*ng*(*A*) и этот объект однозначно восстанавливается по *n*.

Геделевский номер машины Тьюринга

Известно, что любая машина Тьюринга $T = (K, \Sigma, \delta, p_0, p_z, a_0, a_1)$ задается множеством команд вида $p_i a_j \longrightarrow p_k p_l r$, где $p_i,\ p_k \in K$ — состояния, $a_j,\ a_l \in \Sigma$ — символы алфавита ленты, $r \in \{R, L, E\}$ — направление движения головки.

Занумеруем все состояния и символы алфавита натуральными числами: $K \in \{p_0, p_1, \dots, p_z\}$ и $\Sigma \in \{a_0, a_1, \dots, a_s\}$. Будем считать, что состояние p_0 с нулевым номером — начальное и состояние p_z с максимальным номером *z* — заключительное.

Рассмотрим сначала одну команду $p_i a_j \longrightarrow p_k p_l r$. Индексы *i*, *j*, *k*, *l* — это натуральные числа, которые можно записать в какой–либо системе счисления. Выберем двоичную систему счисления, тогда в записи каждой команды используются только следующие символы: *p*, *a*, 0, 1, *R*, *L*, *E*.

Поставим в соответствие каждому символу десятичную цифру:

$$p \to 1,\ a \to 2,\ 0 \to 3,\ 1 \to 4,\ R \to 5,\ L \to 6,\ E \to 7$$

Тогда каждой команде ставится в соответствие целое число в десятичной системе счисления. Например, команде с десятичными индексами $p_3 a_0 \longrightarrow p_5 a_2 R$ или в двоичном эквиваленте $p_{11} a_0 \longrightarrow p_{101} a_{10} R$ соответствует число 1442314342435.

Геделевский номер машины Тьюринга

- любая машина Тьюринга задается набором команд
- каждой команде ставится в соответствие одно число
- всему набору команд — одно длинное число, полученное последовательной записью соответствующих каждой команде чисел.

Для однозначного определения такого длинного числа будем формировать его из отдельных чисел в возрастающей последовательности. Очевидно, что оба требования определения Геделевской нумерации выполняются при предложенном способе кодирования машин Тьюринга в виде натурального числа.

По тезису Тьюринга каждый алгоритм может быть реализован машиной Тьюринга. Возможность нумерации машин Тьюринга означает, что любой вычислимой функции можно поставить в соответствие ее номер.

Возникает вопрос: все ли функции вычислимы? Другими словами, необходимо выяснить существование функций, не вычислимых никакими алгоритмами.

Все ли функции вычислимы?

- Для упрощения рассуждений будем рассматривать только функции одного аргумента. Допустим, что все одноместные функции на множестве натуральных чисел вычислимы. Тогда каждой вычислимой функции можно поставить в соответствие натуральное число — геделевский номер машины Тьюринга, вычисляющей эту функцию.
- Пусть *M* — множество всех вычислимых функций,

$$\mathrm{M} = \{f_0(x), f_1(x), f_2(x), \ldots\}$$

- Построим одноместную функцию *h*(*x*), отличную от всех функций множества *M*. Тем самым мы докажем существование функций, не являющихся вычислимыми.

Все ли функции вычислимы?

- Чтобы сделать построение функции $h(x)$ более наглядным, составим бесконечную матрицу, строками которой будут служить последовательности значений функций $f_0(x), f_1(x), f_2(x), \ldots$, а столбцами — натуральные числа 0,1,2,…, на которых вычисляются значения этих функций:

	0	1	2	…
$f_0(x)$	$f_0(0)$	$f_0(1)$	$f_0(2)$	…
$f_1(x)$	$f_1(0)$	$f_1(1)$	$f_1(2)$	…
$f_2(x)$	$f_2(0)$	$f_2(1)$	$f_2(2)$	…
…	…	…	…	…

- Определим теперь функцию $h(x)$ как функцию, последовательность значений которой получается из последовательности значений, стоящих в нашей таблице на диагонали с увеличением каждого из них на единицу, т.е. $h(a) = f_a(a) + 1$. Эта функция существует в силу нашего конструктивного построения, но функция $h(x)$ не принадлежит множеству M, т.к. она отличается от $f_0(x)$ своим значением для аргумента 0, от $f_1(x)$ своим значением для аргумента 1 и т.д. Другими словами, если $h(x) \in M$, то существует такой натуральный номер m, что для всех x справедливо $h(x) = f_m(x)$. Тогда подставляя вместо переменной x число m получим $h(m) = f_m(m) = f_m(m) + 1$, что невозможно. Полученное противоречие доказывает существование функций, не принадлежащих множеству вычислимых функций.

Проблема остановки машины Тьюринга

- Проблема остановки алгоритма заключается в определении для произвольного алгоритма и произвольных исходных данных, поступающих на вход этому алгоритму, принципа обработки указанным алгоритмом предложенных исходных данных:
  - остановится алгоритм через некоторое конечное число шагов с полученным в процессе работы результатом;
  - не остановится никогда.

Теорема 5.1. Проблема остановки неразрешима.

Доказательство. Допустим, что проблема разрешима. Это значит, что существует алгоритм решения данной проблемы, т.е. существует машина Тьюринга, решающая эту проблему. Эта машина Тьюринга должна получать на вход алгоритм и исходные данные для него и выдавать на выход "да" или "нет" в зависимости от того, остановится или зациклится алгоритм на этих данных. Такая машина Тьюринга должна дейст

$$T_0 : p_0 x * ng(A) \stackrel{*}{\Rightarrow} \begin{cases} p_z 1, & \text{если } A \text{ остановится на } x; \\ p_z \varepsilon, & \text{если } A \text{ не остановится на } x. \end{cases}$$

Легко построить машину Тьюринга, которая копирует исходные данные:

Проблема остановки машины Тьюринга

- $T_{copy} : q_0 u \overset{*}{\Rightarrow} q_z u * u,$

где *u* — цепочка из символов "1".

Поскольку *x* и *ng*(*A*) — натуральные числа, то цепочки *u* и *ng*(*A*) в унарном коде состоят из одних и тех же символов "1". Тогда на вход машины Тьюринга копирования пододадим Геделевский номер *ng*(*A*) и рассмотрим композицию машин Тьюринга

$$T_{copy} \cdot T_0 : q_0 ng(A) \overset{*}{\Rightarrow} q_z ng(A) * ng(A) \overset{*}{\Rightarrow}$$
$$\overset{*}{\Rightarrow} \begin{cases} p_z 1, & \text{если } A \text{ остановится на } ng(A); \\ p_z \varepsilon, & \text{если } A \text{ не остановится на } ng(A). \end{cases}$$

Теперь построим другую машину Тьюринга: $T_1 : t_0 1^n \overset{*}{\Rightarrow} \begin{cases} t_z \varepsilon, & \text{если } n = 0; \\ \infty, & \text{если } n > 0. \end{cases}$

$$T_{copy} \cdot T_0 \cdot T_1 : q_0 ng(A) \overset{*}{\Rightarrow} q_z ng(A) * ng(A) \overset{*}{\Rightarrow}$$

Тогда $\overset{*}{\Rightarrow} \begin{cases} \infty, & \text{если } A \text{ остановится на } ng(A); \\ t_z \varepsilon, & \text{если } A \text{ не остановится на } ng(A). \end{cases}$

Обозначим $T_{res} = T_{copy} * T_0 {}^* T_1$.

Пододадим на вход машины T_{res} ее собственный Геделевский номер $ng(T_{res})$ и получим противоречие:

$$T_{res} : q_0 ng(T_{res}) \overset{*}{\Rightarrow}$$
$$\overset{*}{\Rightarrow} \begin{cases} \infty, & \text{если } T_{res} \text{ остановится на } ng(T_{res}); \\ t_z \varepsilon, & \text{если } T_{res} \text{ не остановится на } ng(T_{res}). \end{cases}$$

Машина Тьюринга T_{res} существовать не может, но $T_{res} = T_{copy} * T_0 {}^* T_1$, причем T_{copy} и T_1 существуют. Следовательно, не существует T_0, и наше предположение о разрешимости проблемы остановки было неверным.

Алгоритмически неразрешимые проблемы

- В силу тезиса Тьюринга невозможность построения машины Тьюринга означает отсутствие алгоритма решения данной проблемы. Поэтому полученная теорема дает первый пример алгоритмически неразрешимой проблемы.

- **Следствие.** Проблема результативности программы неразрешима. Иными словами, не существует общего алгоритма, который для любой программы определил бы, остановится программа при обработке каких-нибудь данных или нет.

Алгоритмически неразрешимые проблемы

- При доказательстве теоремы мы рассматривали машину Тьюринга $T_{copy} * T_0$, которая получает на вход Геделевский номер некоторой машины и определяет, зациклится машина или нет. Алгоритм, который получает на вход свое собственное описание (Геделевский номер) и перерабатывает его, не зацикливаясь, называется самоприменимым.
- Самоприменимыми должны быть трансляторы и интерпретаторы, программы печати текстов программ, и тому подобные программы, исходной информацией для которых также являются программы. Из доказательства теоремы об остановке следует, что машина $T_{copy} * T_0$ существовать не может, следовательно проблема самоприменимости также является неразрешимой.

Алгоритмически неразрешимые проблемы

- **Теорема 5.2 (Теорема Райса).** Никакое нетривиальное свойство вычислимых функций не является алгоритмически разрешимым.

Алгоритмы Маркова

Понятие Марковской подстановки

- Алгоритм *M* исходное слово *P* в алфавите *A* перерабатывает в слово *Q*: *M: P ⇒ Q*
- *Марковская подстановка* - операция над упорядоченной парой слов *(P,Q)*. В заданном слове *R* находят первое вхождение слова *P* (если оно есть) и, не изменяя остальных частей слова *R*, заменяют в нем это вхождение словом *Q*. Полученное слово называется *результатом применения марковской подстановки (P,Q) к слову R*. Если же нет вхождения *P* в слово *R*, то считается, что марковская подстановка *(P,Q)* не

Преобразуемое слово	Марковская подстановка	Результат
541551678	(41551, 00)	500678
шрам	(ра, ар)	шарм
функция	(ε, В-)	В-функция
книга	(кн,ε)	ига
мама	(ε,ε)	мама
дождь	(да,Т)	не применима

Нормальный алгоритм Маркова:

- Запись $P \to Q$ - формула подстановки (P,Q). $P \to Q.$ - формула заключительной подстановки

Упорядоченный конечный список формул подстановок в алфавите A

$$\begin{cases} P_1 \to Q_1(\cdot) \\ P_2 \to Q_2(\cdot) \\ \dots \\ P_n \to Q_n(\cdot) \end{cases}$$

называется ***схемой нормального алгоритма*** в алфавите A.

Последовательность $\{R_i\}$: $R_0 \to R_1 \to R_2 \to \cdots \to R_{m-1} \to R_m$, при $R_0=R$, $R_m=S$ *перерабатывает слово R в слово S.*

Всякий нормальный алгоритм Маркова будет задаваться указанием следующих объектов: алфавита A, схемы нормального алгоритма Z, и, если необходимо, расширения B алфавита A, не имеющего общих букв с алфавитом A.

Алгоритм Маркова работает следующим образом: на каждом шаге применяется правило подстановки с минимально возможным номером. Алгоритм заканчивает работу либо после принятия заключительного правила, либо при отсутствии применимых правил.

Примеры алгоритмов Маркова

Примеры. A={a,b}.
Схема алгоритма

a->ε.
b->b.

Заменяет в слове первое вхождение буквы ‘a’ на пустую цепочку. Если в слове только ‘b’, то оставляет его без изменения.

Заменить последнее вхождение буквы ‘a’ на пустую цепочку.
A={a,b}, B={#,&}

#a->a#
#b->b#
#->&
a&->ε.
b&->&b
&->ε.
ε->#

Вычислимость функций по Маркову

- Функция f, заданная на множестве слов алфавита A, называется ***нормально вычислимой***, если найдется такое расширение B алфавита A и такой нормальный алгоритм в B, что каждое слово P в алфавите A, принадлежащее области определения функции, перерабатывается алгоритмом в f(P).
- Примеры нормально вычислимых функций: x+1, x/3, 2x, (2x+3y)/4.

Принцип нормализации Маркова

всякий алгоритм может быть реализован нормальным алгоритмом Маркова. Или, что эквивалентно, ***всякий алгоритм нормализуем.***

Эквивалентность алгоритмических моделей

Глава 5, стр.121

Эквивалентность алгоритмических моделей

Будем использовать машины Тьюринга в качестве основной модели и докажем эквивалентность всех трех определений алгоритма на основе доказательства четырех теорем:

a) для любой машины Тьюринга можно построить эквивалентную частично–рекурсивную функцию;
b) для любой частично–рекурсивной функции можно построить эквивалентную машину Тьюринга;
c) для любой машины Тьюринга можно построить эквивалентный алгоритм Маркова;
d) для любого алгоритма Маркова можно построить эквивалентную машину Тьюринга.

Следствием из этих теорем является эквивалентность алгоритмов Маркова и частично–рекурсивных функций.

Теорема об эквивалентности машин Тьюринга и частично–рекурсивных функций

Теорема 5.3. Всякая частично–рекурсивная функция вычислима по Тьюрингу.

Доказательство. В соответствии с рекурсивным определением частично–рекурсивной функции для доказательства теоремы достаточно доказать следующие четыре утверждения:

- простейшие функции вычислимы по Тьюрингу;
- с помощью оператора суперпозиции из вычислимых по Тьюрингу функций получаем вычислимые функции;
- с помощью оператора минимизации из вычислимых по Тьюрингу функций получаем вычислимые функции;
- с помощью оператора примитивной рекурсии из вычислимых по Тьюрингу функций получаем вычислимые функции.

Вычислимость простейших функций очевидна, так как легко построить соответствующие машины Тьюринга. Теорема о вычислимости суперпозиции была доказана выше (см. теорему 2.6). Рассмотрим оператор минимизации

Теорема об эквивалентности Машин Тьюринга и частично–рекурсивных функций

Доказательство (продолжение)

Рассмотрим оператор минимизации:

$$f(\bar{x}) = \mu_z(P(\bar{x}, t))$$

Вычисление этой функции можно реализовать с помощью алгоритма

1. $t := 0$;
2. пока $\chi_P(\bar{x}, t) = 0$ вычисляется $t := t + 1$;
3. $f(\bar{x}) = t$

Последовательное выполнение пунктов алгоритма эквивалентно выполнению всего алгоритма при условии существования этих отдельных алгоритмов. Очевидно существование машин Тьюринга T_1 и T_3, где T_1 вычисляет значение $t = 0$ и T_3 возвращает значение t в качестве значения функции $f(\bar{x})$. Второй пункт этого алгоритма представляет собой повторение вычислимых функций по вычислимому предикату. Следовательно, машина Тьюринга T_2, выполняющая такое вычисление, также существует. Тогда существует и композиция $T_1 \cdot T_2 \cdot T_3$.

Вычислимость оператора примитивной рекурсии доказывается аналогично.

Теорема об эквивалентности Машин Тьюринга и частично–рекурсивных функций

Теорема 5.4. Любая вычислимая по Тьюрингу функция является частично–рекурсивной функцией.

Доказательство. Пусть дана произвольная машина Тьюринга

$$T = (K, \Sigma, \delta, p_0, p_z, a_0, a_1)$$

В основу доказательства теоремы положим принцип нумерации поведения машины Тьюринга, заменяя числами как все данные на ленте, так и состояния управляющего устройства, а затем определяя числовые функции, которые характеризуют процессы изменения таких чисел.

Для простоты возьмем в качестве системы счисления число $S = max\{|K|, |\Sigma|\}$.

Заменим символьное определение конфигурации $\langle \alpha, q, a, \beta \rangle$ набором натуральных чисел. Левую часть α цепочки на ленте будем читать как число слева направо, правую часть β цепочки — справа налево. Построим функции, которые в соответствии с командами машины Тьюринга определяют записываемый символ, новое состояние и направление движения, причем знакам направления движения поставим в соответствие, например, следующие числа: $L \to 2$, $R \to 1$, $E \to 0$

Теорема об эквивалентности Машин Тьюринга и частично–рекурсивных функций

Доказательство (продолжение)

Машина Тьюринга на каждом такте выполняет одну какую–либо команду вида $qa \to pbr$. Тогда построим функции $v_q(q,a), v_r(q,a), v_a(q,a)$, которые по текущему состоянию и обозреваемому символу определяют новое состояние, записываемый символ и направление сдвига головки. Каждой команде $qa \to pbr$ можно поставить в соответствие следующее определение

$$v_q(q,a) = p, \quad v_r(q,a) = r = \begin{cases} 0, & r = E \\ 1, & r = R \\ 2, & r = L, \end{cases} \quad v_a(q,a) = b.$$

Множество команд машины Тьюринга *T* конечно по определению, следовательно, данные функции являются примитивно–рекурсивными как функции, полученные с помощью разветвления к примитивно–рекурсивным функциям.

Теорема об эквивалентности Машин Тьюринга и частично–рекурсивных функций

Доказательство (продолжение)

Определим функции на множестве $N \times N \times N \times N$ — множестве четверок чисел $\langle \alpha, q, a, \beta \rangle$, которые являются числовым эквивалентом конфигураций машины Тьюринга. Эти функции определят элементы новой четверки после применения команды на одном такте работы машины Тьюринга:

$$f_\alpha(\alpha, q, a, \beta) = \begin{cases} \alpha, \text{ если } v_r(q,a) = 0 \\ \alpha \cdot S + v_a(q,a), \text{ если } v_r(q,a) = 1 \\ [\alpha / S], \text{ если } v_r(q,a) = 2, \end{cases}$$

$$f_q(\alpha, q, a, \beta) = v_q(q, a),$$

$$f_a(\alpha, q, a, \beta) = \begin{cases} v_a(q,a), \text{ если } v_r(q,a) = 0 \\ \{\beta / S\}, \text{ если } v_r(q,a) = 1 \\ \{\alpha / S\}, \text{ если } v_r(q,a) = 2, \end{cases}$$

$$f_\beta(\alpha, q, a, \beta) = \begin{cases} \beta, \text{ если } v_r(q,a) = 0 \\ [\beta / S], \text{ если } v_r(q,a) = 1 \\ \beta \cdot S + v_a(q,a), \text{ если } v_r(q,a) = 2. \end{cases}$$

Все эти функции, определяющие каждое из новых четырех характеризующих конфигурацию чисел на последующем шаге, — примитивно–рекурсивные.

Теорема об эквивалентности Машин Тьюринга и частично–рекурсивных функций

Доказательство (продолжение)

Рассмотрим теперь поведение машины Тьюринга *T* по тактам и введем функции, являющиеся (кроме уже известных элементов четверки) еще и функцией номера такта *t*:

$$F_\alpha(t,\alpha,q,a,\beta),$$
$$F_a(t,\alpha,q,a,\beta),$$
$$F_q(t,\alpha,q,a,\beta),$$
$$F_\beta(t,\alpha,q,a,\beta).$$

Все эти функции можно определить рекурсивно.

$$F_\alpha(0,\alpha,q,a,\beta)=\alpha,$$
$$F_\alpha(t+1,\alpha,q,a,\beta)=$$
$$=f_\alpha(F_\alpha(t,\alpha,q,a,\beta)),F_q(t,\alpha,q,a,\beta),F_a(t,\alpha,q,a,\beta),F_\beta(t,\alpha,q,a,\beta))$$
$$F_q(0,\alpha,q,a,\beta)=q,$$
$$F_q(t+1,\alpha,q,a,\beta)=$$
$$=f_q(F_\alpha(t,\alpha,q,a,\beta)),F_q(t,\alpha,q,a,\beta),F_a(t,\alpha,q,a,\beta),F_\beta(t,\alpha,q,a,\beta)),$$
$$F_a(0,\alpha,q,a,\beta)=a,$$
$$F_a(t+1,\alpha,q,a,\beta)=$$
$$=f_a(F_\alpha(t,\alpha,q,a,\beta)),F_q(t,\alpha,q,a,\beta),F_a(t,\alpha,q,a,\beta),F_\beta(t,\alpha,q,a,\beta)),$$
$$F_\beta(0,\alpha,q,a,\beta)=\beta,$$
$$F_\beta(t+1,\alpha,q,a,\beta)=$$
$$=f_\beta(F_\alpha(t,\alpha,q,a,\beta)),F_q(t,\alpha,q,a,\beta),F_a(t,\alpha,q,a,\beta),F_\beta(t,\alpha,q,a,\beta)).$$

Теорема об эквивалентности Машин Тьюринга и частично–рекурсивных функций

Доказательство (продолжение)

Приведенные рекурсивные схемы соответствуют так называемой параллельной рекурсии.

Пусть в начальный момент на ленте записана цепочка *x*. В конечный момент

результатом является некоторая цепочка *y*, которая представляет собой функцию от исходных данных: $y = f(x)$.

Рассмотрим эту функцию. В начальный момент при условии правильной работы

$$\alpha = 0;$$
$$q = 0;$$
$$\beta = [z(x)/S], \text{ где } z(x) - \text{зеркальное отображение цепочки } x;$$
$$a = \{z(x)/S.\}$$

Функция зеркального отображения числа является примитивно–рекурсивной в силу следующего определения.

Теорема об эквивалентности Машин Тьюринга и частично–рекурсивных функций

Доказательство (продолжение)

По определению правильной вычислимости перед началом работы машина Тьюринга находится в начальном состоянии ($q = 0$), на ленте имеется исходная цепочка x, головка обозревает первый символ цепочки

$$(\alpha = 0,\ a = \{z(x)/S\},\ \beta = [z(x)/S])$$

Тогда значение $y = f(x)$ определяется при условии правильной вычислимости по формуле

$$y = z(\beta \cdot S + a) = z(S \cdot F_\beta(t_{end}, 0, 0, \{z(x)/S\}, [z(x)/S]) + F_a(t_{end}, 0, 0, \{z(x)/S\}, [z(x)/S])$$

Машина Тьюринга переходит в заключительное состояние и завершает работу на таком шаге t_{end}, когда ее текущее состояние окажется заключительным:

$$t_{end} = \mu_t(q_{end} = F_q(t, 0, 0, \{z(x)/S\}, z[x/S])).$$

Тогда $y = f(x)$ — суперпозиция частично–рекурсивных и примитивно–рекурсивных функций, следовательно, сама является частично–рекурсивной функцией .

Теорема об эквивалентности машин Тьюринга и алгоритмов Маркова

Теорема 5.5. Для произвольной машины Тьюринга можно построить эквивалентный алгоритм Маркова.

Доказательство. Пусть $T = (K, \Sigma, \delta, p_0, p_z, a_0, a_1)$ — произвольная машина Тьюринга. Построим алгоритм Маркова *M*, выполняющий те же действия, что и *T* . В алфавит алгоритма Маркова *M* включим все символы из множества $K \cup \Sigma$. Построим правила *M*. Каждой команде $p_i a_j \longrightarrow p_k a_m r \in \delta$ – машины Тьюринга *T* сопоставим правило или множество правил алгоритма Маркова следующим образом:

а) если *r* = *E*, то правило $p_i a_j \longrightarrow p_k a_m$

б) если *r* = *L*, то для всех $b \in \delta$ включим в множество правил соответствующее правило $bp_i a_j \longrightarrow p_k b a_m$, отмечая тем самым движение головки влево;

в) если *r* = *R*, то для всех $b \in \delta$ включим в множество правил соответствующее правило $p_i a_j b \longrightarrow p_k a_m b$, отмечая тем самым движение головки вправо.

Для того, чтобы начать выполнение алгоритма Маркова в соответствии с правилом, указывающим момент начала работы машины Тьюринга из начального состояния, в котором головка обозревает первый символ исходной цепочки, добавим к множеству правил алгоритма Маркова правило $\varepsilon \rightarrow p_0$ и сделаем это правило последним в списке правил *M*.

Теорема об эквивалентности машин Тьюринга и алгоритмов Маркова

Теорема 5.5. Для произвольной машины Тьюринга можно построить эквивалентный алгоритм Маркова.

Доказательство (продолжение)

Чтобы отметить момент завершения работы машины Тьюринга, добавим к множеству правил *M* заключительное правило с точкой $f \to \varepsilon$.
Осталось реализовать условие "зависания" машины Тьюринга в том случае, когда ее состояние *p* и обозреваемый символ *a* таковы, что в множестве ее команд нет команды с левой частью pa. Это условие также легко реализуется, если множество правил алгоритма Маркова дополнить правилами $pa \to pa$ для всех $p \in K$ и $a \in \Sigma$, для которых в δ отсутствует команда с левой частью pa.

Эквивалентность исходной машины Тьюринга *T* и построенного алгоритма Маркова *M* очевидна.

Теорема об эквивалентности машин Тьюринга и алгоритмов Маркова

Теорема 5.6. Для произвольного алгоритма Маркова можно построить эквивалентную машину Тьюринга.

Доказательство. Пусть $M = \{U\}$ — алгоритм Маркова над заданным алфавитом δ. Рассмотрим следующие вспомогательные машины Тьюринга.

$T_{\alpha i}$ — поиск цепочки α_i в текущей цепочке, которая имеется на ленте. Если цепочка найдена, машина Тьюринга $T_{\alpha i}$ переходит в состояние p_i и обозревает последний символ найденной цепочки α_i, в противном случае она переходит в состояние f_i. Эти два состояния нам в дальнейшем понадобятся для реализации разветвления к различным машинам Тьюринга.

$T^{l}_{\beta i}$ — заменить на левой полуленте цепочку α_i на цепочку β_i. Перед началом работы машина Тьюринга обозревает последний символ цепочки α_i. Заменив цепочку α_i на β_i, машина Тьюринга переходит в заключительное состояние и обозревает последний символ цепочкиβ_i.

T_{begin} — перейти к началу цепочки, записанной на ленте.

T_{mar}— разделить ленту на полуленты. Неподвижный маркер ставится справа от обозреваемой ячейки.

T_{del} — удалить маркер, разделяющий ленту на полуленты и слить содержимое полулент.

T_{end}— закончить работу, для чего необходимо вернуть головку к началу цепочки и прейти в заключительное состояние.

Теорема об эквивалентности машин Тьюринга и алгоритмов Маркова

Доказательство (продолжение)

Построим теперь из вспомогательных машин Тьюринга несколько новых:
а) если правило $\alpha \to \beta$ алгоритма Маркова было не заключительным, то

$$T_{i,1} = T_{mar} \cdot T_{\beta i} \cdot T_{del} \cdot T_{begin}$$

в соответствии с тем, как были определены вспомогательные машины Тьюринга, эта машина Тьюринга заменит цепочку α на β и вернется к началу данных на ленте;

если правило было заключительным, то

$$T_{i,1} = T_{mar} \cdot T_{\beta} \cdot T_{del} \cdot T_{begin} \cdot T_{end}$$

в отличие от предшествующего варианта данная машина Тьюринга выполнит те же действия и закончит работу;

а) рассмотрим машину Тьюринга $T_{i,2}$:

$$T_{i,2} = T_{\alpha i} \begin{matrix} p_i \nearrow T_{i,1} \\ f_i \searrow T_{i+1,2} \end{matrix}$$

Теорема об эквивалентности машин Тьюринга и алгоритмов Маркова

Доказательство (продолжение)

Осталось теперь определить машину Тьюринга $T_{n+1,2}$ где $n = \{U\}$ — число правил алгоритма Маркова $\mathrm{M} = \{U\}$. Машина $T_{n+1,2}$ начнет работу только при условии, что ни одно из правил множества *U* не найдено на ленте. Это означает, что алгоритм Маркова должен закончить свою работу.

Эквивалентное поведение построенной машины Тьюринга будет при условии, что $T_{n+1,2} = T_{end}$.

Машины Тьюринга $T_{i,1}$ и $T_{i,2}$ существуют, т.к. они построены с помощью композиции и разветвления из существующих машин Тьюринга. Очевидно, что построенная нами машина Тьюринга $T_{1,2}$ выполняет те же преобразования исходной цепочки, что и алгоритм Маркова *M*.

Следствие. Определения алгоритма в виде машины Тьюринга, алгоритма Маркова и частично–рекурсивной функции эквивалентны.