4 АЛГОРИТМЫ

4.1 Программы

В дальнейшем мы будем рассматривать сигнатуру Ω_0, состоящую из символов операций одноместного $'$ и нульместного 0 и символа двухместного отношения $<$. Слабой арифметикой называется алгебраическая система сигнатуры Ω_0, основное множество которой — множество натуральных чисел, а $'$ — это операция прибавления единицы. 0 и $<$ в арифметике обозначают наименьший элемент и обычное отношение порядка на натуральных числах.

Под состоянием будем понимать состояние слабой арифметики, другими словами, отображение, которое каждой переменной приписывает в качестве значения натуральное число.

Определение 4.1.1 (присваивания). *Присваиванием называется каждое выражение одного из видов:*

а). $x \leftarrow x'$,

б). $x \leftarrow 0$,

в). $x \leftarrow y$,

где x,y — переменные. В присваиваниях видов а) и б) используется переменная x, а в присваивании вида в) — переменные x и y.

Определение 4.1.2 (условия). *Условием называется каждое выражение одного из видов:*

а). $x = y$,

б). $x < y$,

где x,y — переменные.

В условиях видов а) и б) используются переменные x и y.

Определение 4.1.3 (теста). *Тестом называется каждое выражение вида ϕ?, где ϕ — условие.*

Определение 4.1.4 (структурированной программы).

а). Каждое присваивание является структурированной программой.

б). Если Π_1 и Π_2 — структурированные программы, то

"$\Pi_1;\Pi_2$ —

тоже структурированная программа, называемая композицией программ Π_1 и Π_2. В ней используются переменные, которые или используются в Π_1, или используются в Π_2.

в). Если Π_1, Π_2 *— структурированные программы, а* ϕ *— условие, то*

"если ϕ *то* Π_1 *иначе* Π_2 *конец "*

тоже есть структурированная программа, называемая условным оператором. В ней используются переменные, используемые в ϕ, *в* Π_1 *или в* Π_2.

г). Если Π_1 *— структурированная программа, а* ϕ *— условие, то*

"пока ϕ *делай* Π_1 *все "*

тоже есть структурированная программа, называемая циклом с телом Π_1. *В ней используются переменные, используемые в* ϕ *или в* Π_1.

д). Каждая вещь является структурированной программой, только в случае, когда это можно доказать, используя а)—г).

Определение 4.1.5 (работы программы на состоянии). *Пусть* σ *является состоянием.*

а). $(x \leftarrow x')(\sigma)$ *есть такое состояние* σ_1, *что* $\sigma_1(y) = \sigma(y)$ *для всех переменных* y, *отличных от* x, *и* $\sigma_1(x) = (\sigma(x))'$.

б). $(x \leftarrow 0)(\sigma)$ *есть такое состояние* σ_1, *что* $\sigma_1(y) = \sigma(y)$ *для всех переменных* y, *отличных от* x, *и* $\sigma_1(x) = 0$.

в). $(x \leftarrow y)(\sigma)$ *есть такое состояние* σ_1, *что* $\sigma_1(z) = \sigma(z)$ *для всех переменных* z, *отличных от* x, *и* $\sigma_1(x) = \sigma(y)$.

г). $(\Pi_1;\Pi_2)(\sigma) = \Pi_2(\Pi_1(\sigma))$. *Если* $\Pi_1(\sigma)$ *не определено или если* $\Pi_2(\sigma_1)$ *не определено для* $\sigma_1 = \Pi_1(\sigma)$, *то* $(\Pi_1;\Pi_2)(\sigma)$ *не определено.*

д). Пусть Π *есть*

"если $x = y$ *то* Π_1 *иначе* Π_2 *конец "*

Тогда

$$\Pi(\sigma) = \begin{cases} \Pi_1(\sigma), & \text{если } \sigma(x) = \sigma(y), \\ \Pi_2(\sigma), & \text{в остальных случаях.} \end{cases}$$

В случае, когда $\sigma(x) = \sigma(y)$, $\Pi(\sigma)$ *не определено, если* $\Pi_1(\sigma)$ *не определено. В случае, когда* $\sigma(x) \neq \sigma(y)$, $\Pi(\sigma)$ *не определено, когда* $\Pi_2(\sigma)$ *не определено.*

Пусть Π *есть*

"если $x < y$ *то* Π_1 *иначе* Π_2 *конец "*

Тогда

$$\Pi(\sigma) = \begin{cases} \Pi_1(\sigma), & \text{если } \sigma(x) < \sigma(y), \\ \Pi_2(\sigma), & \text{в остальных случаях.} \end{cases}$$

В случае, когда $\sigma(x) < \sigma(y)$, $\Pi(\sigma)$ *не определено, если* $\Pi_1(\sigma)$ *не определено. В случае, когда* $\sigma(x) \geq \sigma(y)$, $\Pi(\sigma)$ *не определено, когда* $\Pi_2(\sigma)$ *не определено.*

е). ("пока $x = y$ делай Π_1 все ")(σ) есть такое состояние σ_m, что либо $\sigma(y) \neq \sigma(x)$ и $\sigma_m = \sigma$, либо $\sigma_0 = \sigma$, $\sigma_1 = \Pi_1(\sigma_0), \dots, \sigma_m = \Pi_1(\sigma_{m-1})$ определены, $\sigma_i(x) = \sigma_i(y)$ для $i = 0, 1, \dots, m-1$, но $\sigma_m(y) \neq \sigma_m(x)$. Если такого состояния σ_m нет, то рассматриваемый цикл не определен на σ.

("пока $x < y$ делай Π_1 все ")(σ) есть такое состояние σ_m, что либо $\sigma(y) \leq \sigma(x)$ и $\sigma_m = \sigma$, либо $\sigma_0 = \sigma$, $\sigma_1 = \Pi_1(\sigma_0), \dots, \sigma_m = \Pi_1(\sigma_{m-1})$ определены, $\sigma_i(x) < \sigma_i(y)$ для $i = 0, 1, \dots, m-1$, но $\sigma_m(y) \leq \sigma_m(x)$. Если такого состояния σ_m нет, то рассматриваемый цикл не определен на σ.

Если $\Pi(\sigma)$ не определено, то говорят, что структурированная программа Π не останавливается на состоянии σ.

Менее формально, присваивание видов а), б) и в) меняет только значение x и не меняет значений других переменных. Новое значение x есть увеличенное на 1 старое значение x, либо 0, либо старое значение y соответственно. Выполнить композицию программ означает последовательно выполнить эти программы. Выполнить условный оператор означает выполнить первую из его программ, если условие истинно, и вторую, если оно ложно. Выполнить цикл означает выполнять тело цикла до тех пор, пока условие истинно.

Заметим еще, что композиция программ $\Pi_1; \Pi_2$ и Π_3 совпадает с композицией программ Π_1 и $\Pi_2; \Pi_3$.

Действительно, работа обеих композиций одинакова:

$$((\Pi_1; \Pi_2); \Pi_3)(\sigma) = \Pi_3((\Pi_1; \Pi_2)(\sigma)) = \Pi_3(\Pi_2(\Pi_1(\sigma))),$$
$$(\Pi_1; (\Pi_2; \Pi_3))(\sigma) = (\Pi_2; \Pi_3)(\Pi_1(\sigma)) = \Pi_3(\Pi_2(\Pi_1(\sigma))).$$

Это позволяет использовать записи вида $\Pi_1; \Pi_2; \Pi_3$.

Легко проверяется, что если Π не использует переменной x и $\sigma_1 = \Pi(\sigma)$, то $\sigma_1(x) = \sigma(x)$.

Структурированные программы предназначены для вычисления арифметических функций. Напомним, что через ω мы обозначаем множество $\{0, 1, 2, \dots\}$ натуральных чисел. Арифметической функцией называют любое отображение подмножества множества ω^n в ω для фиксированного n из ω. При этом n называют местностью (или числом аргументов) функции. Арифметическая функция ϕ местности n отображает подмножество M из ω^n в ω. Если $\langle a_1, \dots, a_n \rangle$ лежит в M, то говорят что $\phi(a_1, \dots, a_n)$ определено. Если $M = \omega$, функция ϕ называется всюду определенной. Если M пусто, функция ϕ называется нигде не определенной.

Определение 4.1.6 (функции, вычислимой программой). *Пусть* Π *— структурированная программа, использованные переменные которой разбиты на основные* $x_1, \ldots, x_n$ *и рабочие. Пусть* σ *— такое состояние, которое всем рабочим переменным присваивает значение 0. Пусть* $\sigma(x_1) = a_1, \ldots, \sigma(x_n) = a_n$*. Пусть* x *— используемая* Π *переменная. Тогда* $\phi_{\Pi,x}(a_1, \ldots, a_n) = (\Pi(\sigma))(x)$*. Если* $\Pi(\sigma)$ *не определено, то* $\phi_{\Pi,x}(a_1, \ldots, a_n)$ *тоже не определено.* $\phi_{\Pi,x}$ *называется функцией, вычисляемой* Π *в* x *при указанном разбиении переменных. Функция называется программно вычислимой, если она вычисляется некоторой* Π *в некоторой переменной* x *при некотором разбиении используемых* Π *переменных на основные и рабочие.*

Менее формально, $\phi_{\Pi,x}$ — это заключительное значение $(\Pi(\sigma))(x)$ для такого состояния, которое присваивает 0 всем рабочим переменным. Если же $\Pi(\sigma)$ не определено, то $\phi_{\Pi,x}$ на соответствующем наборе тоже не определена.

Приведем несколько примеров.

Пример 4.1.1.

Рассмотрим следующую программу Π_1:

пока $z < y$ делай $z \leftarrow z'$; $x \leftarrow x'$ все.

Разобьем используемые Π_1 переменные на основные x, y и рабочую z.

Покажем, что Π_1 в x вычисляет $x + y$.

Заметим, что начальное значение z есть 0. При начальном значении y равном 0 тело цикла не выполняется ни разу и заключительное значение x совпадает с начальным. Но $x+0$ тоже есть x. В остальных случаях тело цикла выполняется y раз и поэтому значение x увеличится на y.

Пример 4.1.2.

В следующей программе Π_2

$x \leftarrow x'$

используется только одна переменная x, которая считается основной. Эта программа в x вычисляет $x + 1$.

Пример 4.1.3.

Программа Π_3

$x \leftarrow 0$

в основной переменной x вычисляет функцию одного аргумента, равную 0 при всех значениях этого аргумента.

Пример 4.1.4.

Для каждых n и i рассмотрим программу $\Pi_4(n,i)$
$x_1 \leftarrow x_1; \ldots; x_n \leftarrow x_n;\ x_1 \leftarrow x_i$
Считая переменные $x_1, \ldots, x_n$ основными, получим, что эта программа в x_1 вычисляет функцию от n аргументов, при любом наборе их значений равную i-тому из них.

Определение 4.1.7 (характеристической функции). *Пусть M — подмножество ω^n. Арифметическая функция ϕ местности n называется характеристической функцией множества M, если*

$$\phi(x_1, \ldots, x_n) = \begin{cases} 1, & \text{если } \langle x_1, \ldots, x_n \rangle \in M, \\ 0, & \text{в остальных случаях.} \end{cases}$$

Характеристическая функция множества M обозначается через χ_M.

Определение 4.1.8. *Пусть M — подмножество ω^n. Говорят, что существует алгоритм для определения принадлежности последовательности длины n натуральных чисел к M, если характеристическая функция множества M программно вычислима.*

Определение 4.1.9. *Нумерацией структурированных программ назовем такое отображение, которое каждой структурированной программе Π так ставит в соответствие непустое множество натуральных чисел — номеров этой программы, что при этом никакое натуральное число не является номером двух разных программ.*

Зафиксируем некоторую переменную x. Если программа использует x, отнесем x к основным, а все остальные переменные, используемые рассматриваемой программой, к рабочим. Если для структурированной программы, использующей x, функция $\phi_{\Pi,x}(m)$ определена, то скажем, что программа Π останавливается на m.

Теорема 4.1. *Какова бы ни была нумерация структурированных программ, не существует алгоритма, определяющего по натуральному числу, является ли это число номером структурированной программы, останавливающейся на этом своем номере.*

Доказательство. От противного. Пусть такой алгоритм существует. Тогда программно вычислима функция

$$\phi(x) = \begin{cases} 1, & \text{если } x \in M, \\ 0, & \text{в остальных случаях.} \end{cases}$$

Через M здесь обозначено множество всех номеров структурированных программ, останавливающихся на этом своем номере. Другими словами, $x \in M$ тогда и только тогда, когда программа номера x останавливается на x. Пусть эта функция вычисляется структурированной программой Π в x. Можно считать, что x — единственная основная переменная этой программы.

Пусть переменная y отлична от x. Рассмотрим программу Π':

Π; $y \leftarrow 0$; пока $y < x$ делай $x \leftarrow x$ все

Если n — номер структурированной программы, останавливающейся на этом своем номере, то $\phi_{\Pi,x}(n)$ есть 1. Поэтому $\phi_{\Pi',x}(n)$ не определено. В остальных случаях $\phi_{\Pi,x}(n)$ есть 0 и $\phi_{\Pi',x}(n) = \phi_{\Pi,x}(n) = 0$. Пусть m — номер Π'. Если Π' останавливается на своем номере m, то $\phi(m) = 1$ и, значит, $\phi_{\Pi,x}(m) = 1$. Но, по предыдущему, тогда $\phi_{\Pi',x}(m)$ не определено. Значит, программа Π' не останавливается на m и этот случай не имеет места. Однако, если Π' не останавливается на своем номере m, то $\phi(m) = 0$ и, значит, $\phi_{\Pi,x}(m) = 0$. Но, по предыдущему, тогда $\phi_{\Pi',x}(m) = 0$ и Π' останавливается на своем номере m. Опять противоречие.

Итак, функция ϕ не является программно вычислимой. ∎

Заметим, что совершенно не важно, каким способом занумерованы программы. Каждая программа может иметь даже много номеров. В доказательстве используется лишь то, что каждая программа имеет хотя бы один номер, а каждое натуральное число является номером не более одной программы.

Определение 4.1.10. *Нумерацию структурированных программ назовем правильной, если программно вычислима некоторая всюду определенная функция $\chi(x)$, удовлетворяющая следующему условию:*

если n — номер структурированной программы Π, то $\chi(n)$ — это один из номеров структурированной программы

$$\underbrace{x \leftarrow x'; \ldots; x \leftarrow x'}_{n\ \text{раз}}; \Pi$$

если же n не является номером никакой структурированной программы, то $\chi(n)$ тоже не является номером никакой структурированной программы.

Теорема 4.2. *Какова бы ни была правильная нумерация структурированных программ, не существует алгоритма, определяющего по натуральному числу, является ли это число номером структурированной программы, останавливающейся на 0.*

Доказательство. От противного. Пусть такой алгоритм существует. Тогда программно вычислима функция

$$\phi(x) = \begin{cases} 1, & \text{если } x \in M_0, \\ 0, & \text{в остальных случаях.} \end{cases}$$

Через M_0 здесь обозначено множество всех номеров структурированных программ, останавливающихся на 0. Другими словами, $x \in M_0$ тогда и только тогда, когда программа номера x останавливается на 0.

Рассмотрим структурированную программу P, вычисляющую ϕ в x. Можно считать, что x — единственная основная переменная этой программы. Рассмотрим также структурированную программу P_1, вычисляющую χ в x с единственной основной переменной x. Пусть $y_1, \ldots, y_k$ составляют список всех отличных от x переменных, используемых программой P_1. Можно считать, что эти переменные не используются программой P.

Если m — номер структурированной программы Π, останавливающейся на этом своем номере, то программа

$$\underbrace{x \leftarrow x'; \ldots; x \leftarrow x';}_{m \text{ раз}} \Pi \tag{12}$$

останавливается на 0. Так как $\chi(m)$ дает один из номеров программы (12), то $\phi_{P,x}(\phi_{P_1,x}(m))$ есть 1, так как программа номера $\phi_{P_1,x}(m)$ останавливается на 0.

Если же m не есть номер структурированной программы, останавливающейся на этом своем номере, то $\phi_{P,x}(\phi_{P_1,x}(m)) = 0$, так как либо программа номера $\phi_{P_1,x}(m)$ не останавливается на 0, либо $\phi_{P_1,x}(m)$ не является номером никакой структурированной программы. Но

$$\phi_{P,x}(\phi_{P_1,x}(m)) = \phi_{P_1;P,\ x}(m).$$

Поэтому $P_1; P$ вычисляет характеристическую функцию ϕ из доказательства теоремы 4.1, что противоречит теореме 4.1. ∎

Некоторая конкретная нумерация структурированных программ будет построена позже.

4.2 Рекурсивные функции

Определение 4.2.1 (суперпозиции арифметических функций). *Пусть заданы арифметические функции ϕ от m аргументов и $\phi_1, \ldots,$*

ϕ_m, каждая от n аргументов. Определяется функция $[\phi;\ \phi_1, \ldots, \phi_m]$ от n аргументов следующим образом.

Эта функция определена на наборе $\langle a_1, \ldots, a_n \rangle$ только в случае, когда

$$\phi_1(a_1, \ldots, a_n), \ldots, \phi_m(a_1, \ldots, a_n)$$

определены и

$$\phi(\phi_1(a_1, \ldots, a_n), \ldots, \phi_m(a_1, \ldots, a_n))$$

тоже определено. В этом случае

$$[\phi;\ \phi_1, \ldots, \phi_m](a_1, \ldots, a_n)$$

есть

$$\phi(\phi_1(a_1, \ldots, a_n), \ldots, \phi_m(a_1, \ldots, a_n)).$$

Определение 4.2.2 (примитивной рекурсии). *Пусть заданы арифметические функции ϕ от n аргументов и χ от $n+2$ аргументов.*

Определяется функция $R(\phi, \chi)$ от $n+1$ аргумента следующим образом.

Эта функция определена на наборе $\langle a_1, \ldots, a_n, 0 \rangle$ только в случае, когда ϕ определена на наборе $\langle a_1, \ldots, a_n \rangle$ и тогда $(R(\phi, \chi))(a_1, \ldots, a_n, 0)$ есть $\phi(a_1, \ldots, a_n)$.

Эта функция $R(\phi, \chi)$ определена на наборе $\langle a_1, \ldots, a_n, b{+}1 \rangle$ только в случае, когда $(R(\phi, \chi))(a_1, \ldots, a_n, b)$ определено и χ определена на наборе

$$\langle a_1, \ldots, a_n, b, (R(\phi, \chi))(a_1, \ldots, a_n, b) \rangle,$$

и тогда

$$(R(\phi, \chi))(a_1, \ldots, a_n, b+1)$$

есть

$$\chi(a_1, \ldots, a_n, b, (R(\phi, \chi))(a_1, \ldots, a_n, b)).$$

Пример 4.2.1.

Функция $+$ получается как $R(\eta_1^1, [';\ \eta_3^3])$, где η_1^1 — функция от одного аргумента, равная своему аргументу, η_3^3 — функция от трех аргументов, равная третьему из них, $'$ — функция прибавления единицы из § 4.1.

Действительно, $a+0 = a$ и $a+(b+1) = (a+b)+1 = \eta_3^3(a, b, a+b)+1$.

Пример 4.2.2.

Функция $*$ (умножение) получается как $R(\theta, [+;\ \eta^3_1, \eta^3_3])$, где θ — функция от одного аргумента, равная 0 при любом значении аргумента.

Действительно,

$a * 0 = 0$; $a * (b+1) = a * b + a = \eta^3_3(a, b, a * b) + \eta^3_1(a, b, a * b)$.

Пример 4.2.3.

Пусть $(-:)(0)$ есть 0 и $(-:)(b{+}1)$ есть b. Рассмотрим такую функцию ϕ от двух аргументов, что $\phi(a, b)$ есть $(-:)(b)$.

Тогда $\phi(a, 0)$ есть 0, что равно $\theta(a)$, а

$$\phi(a, b+1) = b = \eta^3_2(a, b, \phi(a, b)).$$

Поэтому $\phi = R(\theta, \eta^3_2)$. Однако,

$$(-:) = [\phi;\ \eta^1_1,\ \eta^1_1] = [R(\theta,\ \eta^3_2);\ \eta^1_1,\ \eta^1_1].$$

Пусть $a \frown b = 0$, если $a \le b$, и $a \frown b$ есть обычная разность a и b, если $a > b$.

Тогда $a \frown 0 = a$ и $a \frown (b+1) = (-:)(a \frown b) = [(-:); \eta^3_3](a, b, a \frown b)$. Значит,

$$\frown = R(\eta^1_1, [(-:); \eta^3_3]) = R(\eta^1_1, [[R(\theta, \eta^3_2);\ \eta^1_1,\ \eta^1_1]; \eta^3_3]).$$

Определение 4.2.3 (минимизации). *Пусть задана функция ϕ от $n+1$ аргумента. Определяется функция $M\phi$ от n аргументов следующим образом. $(M\phi)(a_1, \ldots, a_n)$ равно i, если $i = 0$ и $\phi(a_1, \ldots, a_n, 0)$ определено и равно 0, либо если $i > 0$, $\phi(a_1, \ldots, a_n, 0), \ldots, \phi(a_1, \ldots, a_n, i-1)$ определены и отличны от 0, но $\phi(a_1, \ldots, a_n, i)$ определено и равно 0.*

Менее формально, $(M\phi)$ определена на наборе $\langle a_1, \ldots, a_n\rangle$ и равна i, если i — наименьший корень уравнения $\phi(a_1, \ldots, a_n, y) = 0$ и ϕ определена на всех наборах

$$\langle a_1, \ldots, a_n, 0\rangle, \ldots, \langle a_1, \ldots, a_n, i-1\rangle.$$

Пример 4.2.4.

Целая часть $[:](x, y)$ от деления x на y при y отличном от 0 — это наименьшее из таких z, что $y * (z+1) > x$. Так как $y * (z+1) > x$ эквивалентно $y * (z+1) \ge x+1$ и эквивалентно $(x+1) \frown (y * (z+1)) = 0$, то $[:]$ есть

$$M([\frown; [';\ \eta^3_1], [*;\ \eta^3_2, [';\ \eta^3_3]]]).$$

При таком определении $[:]$ не определено при $y = 0$ для любого x.

Пример 4.2.5.

$M([';\ +])$ — это нигде не определенная функция,так как $x + y + 1$ всегда больше 0.

Определение 4.2.4 (простейшей функции). *Функция называется простейшей, если она есть $'$ (имеет один аргумент и значение функции получается прибавлением 1 к значению этого аргумента), θ (всюду определенная функция одного аргумента, равная 0 при любом значении своего аргумента), либо при некоторых таких натуральных положительных i и j, что $j \leq i$, есть η^i_j (всюду определенная функция от i аргументов, равная j-тому из них).*

Более подробно, $\eta^i_j(a_1, \dots, a_i) = a_j$ для любых натуральных $a_1, \dots, a_i$.

Определение 4.2.5 (частично рекурсивной функции). *Арифметическая функция ϕ называется частично рекурсивной, если существует такая конечная последовательность $\phi_1, \dots, \phi_m$ арифметических функций, что каждая функция этой последовательности либо является простейшей, либо получается из предыдущих функций этой же последовательности при помощи суперпозиции, примитивной рекурсии или минимизации, а последняя функция ϕ_m этой последовательности есть ϕ. Скажем, что последовательность $\phi_1, \dots, \phi_m$ задает частично рекурсивную функцию ϕ.*

Замечание 4.2.1. *Функция тогда и только тогда является частично рекурсивной, когда она представляется в виде терма, правильно построенного из символов $'$, θ, η^i_j, R, $[\ \ ]$ и M.*

Действительно, индукцией по числу символов M, $[\ \ ]$ и R в таком терме показываем, что он задает частично рекурсивную функцию. Наоборот, индукцией по длине m последовательности арифметических функций, задающих частично рекурсивную функцию ϕ, доказываем, что ϕ задается термом рассматриваемого вида.

Менее формально, арифметическая функция частично рекурсивна тогда и только тогда, когда она получается из простейших конечным числом суперпозиций, примитивных рекурсий и минимизаций.

Всюду определенная частично рекурсивная функция называется рекурсивной. Из примеров видно, что функции $+$, $*$, $\frown$ являются рекурсивными.

Пример 4.2.6.

Арифметическая функция sg определяется так: $sg(x)$ есть 1, если $x > 0$, и $sg(0) = 0$. Функция $\overline{sg}$ есть $[\frown; [';\ \theta], sg]$, другими словами, $\overline{sg}(x)$ есть $1 \frown sg(x)$. Значит, $\overline{sg}(0) = 1$ и $\overline{sg}(x) = 0$ для $x > 0$.

Рассмотрим функцию $\phi(x, y) = sg(y)$. Ясно, что ϕ есть $R(\theta, [';[\theta;\ \eta_3^3]])$. Так как $sg = [\phi;\ \eta_1^1,\ \eta_1^1]$, то sg и $\overline{sg}$ — рекурсивные функции.

Пример 4.2.7.

Пусть $|a - b|$ есть $a \frown b$, если $a \geq b$, и $b \frown a$, если $a < b$. Эту функцию назовем модулем разности. Заметим, что

$$|a - b| = (a \frown b) + (b \frown a).$$

Поэтому модуль разности — рекурсивная функция.

Определение 4.2.6 (ограниченного суммирования).

$$\sum_{i=0}^{y} \phi(x_1, \ldots, x_n, i)$$

для заданной всюду определенной арифметической функции ϕ от $n+1$ аргумента есть такая всюду определенная арифметическая функция χ от $n+1$ аргумента, что

$$\chi(a_1, \ldots, a_n, b) = \phi(a_1, \ldots, a_n, 0) + \cdots + \phi(a_1, \ldots, a_n, b)$$

для любого набора $\langle a_1, \ldots, a_n, b\rangle$ натуральных чисел. Говорят, что χ получается из ϕ ограниченным суммированием.

Определение 4.2.7 (ограниченного перемножения).

$$\prod_{i=0}^{y} \phi(x_1, \ldots, x_n, i)$$

для заданной всюду определенной арифметической функции ϕ от $n+1$ аргумента есть такая всюду определенная арифметическая функция χ от $n+1$ аргумента, что

$$\chi(a_1, \ldots, a_n, b) = \phi(a_1, \ldots, a_n, 0) * \cdots * \phi(a_1, \ldots, a_n, b)$$

для любого набора $\langle a_1, \ldots, a_n, b\rangle$ натуральных чисел. Говорят, что χ получается из ϕ ограниченным перемножением.

Теорема 4.3. *Если функция получается из рекурсивной ограниченным суммированием или перемножением, то она тоже рекурсивна.*

Доказательство. Только для суммирования. Для перемножения оно аналогично.

Пусть χ получается из ϕ ограниченным суммированием. Тогда

$$\chi(a_1,\dots,a_n,0)=\phi(a_1,\dots,a_n,0)=\\ [\phi;\ \eta_1^n,\dots,\eta_n^n,[\theta;\ \eta_1^n]](a_1,\dots,a_n);$$

$$\chi(a_1,\dots,a_n,b+1)=\chi(a_1,\dots,a_n,b)+\phi(a_1,\dots,a_n,b+1)=\\ [+;\ \eta_{n+2}^{n+2},[\phi;\ \eta_1^{n+2},\dots,\eta_n^{n+2},[';\ \eta_{n+1}^{n+2}]]](a_1,\dots,a_n,b,\chi(a_1,\dots,a_n,b)).$$

■

Определение 4.2.8 (разбора случаев). *Пусть заданы всюду определенные арифметические функции $\phi_1,\dots,\phi_m$, ϕ_{m+1} и $\chi_1,\dots,\chi_m$ от n аргументов. Пусть выполнено следующее условие совместности:*

не существует таких i и j из $\{1,\dots,m\}$ и такого набора $\langle a_1,\dots,a_n\rangle$ натуральных чисел, что $i<j$, $\chi_i(a_1,\dots,a_n)=0$ и $\chi_j(a_1,\dots,a_n)=0$ одновременно.

Пусть на наборе $\langle a_1,\dots,a_n\rangle$ натуральных чисел функция ϕ равна $\phi_i(a_1,\dots,a_n)$, если $\chi_i(a_1,\dots,a_n)=0$ и i из $\{1,\dots,m\}$. Если же на этом наборе все функции $\chi_1,\dots,\chi_m$ отличны от 0, то ϕ на этом наборе равна $\phi_{m+1}(a_1,\dots,a_n)$. Говорят, что эта всюду определенная арифметическая функция ϕ от n аргументов получается из функций $\phi_1,\dots,\phi_m$, ϕ_{m+1}, $\chi_1,\dots,\chi_m$ разбором случаев.

Теорема 4.4. *Если функция ϕ получается из рекурсивных n-местных функций $\phi_1,\dots,\phi_m$, ϕ_{m+1}, $\chi_1,\dots,\chi_m$, для которых выполнено условие совместности, разбором случаев, то ϕ тоже рекурсивна.*

Доказательство. Следует из равенства

$$\phi(a_1,\dots,a_n)=\phi_1(a_1,\dots,a_n)*\overline{sg}(\chi_1(a_1,\dots,a_n))+\dots+\\ \phi_m(a_1,\dots,a_n)*\overline{sg}(\chi_m(a_1,\dots,a_n))+\\ \phi_{m+1}(a_1,\dots,a_n)*sg(\chi_1(a_1,\dots,a_n)*\dots*\chi_m(a_1,\dots,a_n)).$$

■

Пример 4.2.8.

Обозначим для натуральных x и y через $rest(x,y)$ остаток от деления x на y, считая пока y отличным от 0.

Рассмотрим функцию $\sum_{i=0}^{z} \overline{sg}((i*y)\frown x)$. Эта функция всюду определенная. Обозначим ее через $\phi(x,y,z)$. По теореме 4.3, эта функция ϕ рекурсивна. Поэтому рекурсивна и функция $\chi(x,y) = \phi(x,y,x)$. Ясно, что $\chi(x,y) = \sum_{i=0}^{x} \overline{sg}((i*y)\frown x)$. Когда y отлично от 0 и $i \le [:](x,y)$, то $i*y \le x$ и $\overline{sg}((i*y)\frown x) = 1$. Если же $i > [:](x,y)$, то $i*y > x$ и $\overline{sg}((i*y)\frown x) = 0$. Значит, при y, отличном от 0, $\chi(x,y) = [:](x,y)+1$. Например, $\chi(0,y) = \sum_{i=0}^{0} \overline{sg}(0) = 1$. Поэтому при y, отличном от 0, $[:](x,y) = \chi(x,y)\frown 1$. Функция $\chi(x,y)\frown 1$ рекурсивна и, в частности, определена и при $y=0$. Заметим, что $\chi(x,0)\frown 1 = x$. Нам удобно положить $[:](x,0) = x$. При таком доопределении $[:](x,y)$ — рекурсивная функция.

При y, отличном от 0, $rest(x,y) = x \frown (y*[:](x,y))$. При $y = 0$ пусть это равенство определяет $rest(x,y)$. Тогда $rest(x,0) = x$. Теперь $rest(x,y)$ — рекурсивная функция.

Пример 4.2.9.

Натуральное число называется простым, если оно имеет точно два делителя. Если $\phi_{\text{Пр}}$ — характеристическая функция множества Пр простых чисел, то

$$\phi_{\text{Пр}}(x) = \overline{sg}(|(\sum_{i=0}^{x} \overline{sg}(rest(x,i))) - 2|).$$

Из этого равенства видно, что $\phi_{\text{Пр}}$ — рекурсивная функция.

Пример 4.2.10.

Известно, что множество простых чисел бесконечно. Пусть $p(i)$ — это i-тое по порядку простое число, если самое маленькое простое число считать нулевым по порядку. В частности, $p(0) = 2$, $p(1) = 3$, $p(2) = 5$, $p(3) = 7$, $p(4) = 11$ и так далее. Ясно, что $p(n)$ — это наименьшее из таких m, что среди чисел $0, 1, \ldots, m$ имеется $n+1$ простых чисел.

Число $\phi(m)$ простых чисел среди $0, 1, \ldots, m$ определяется формулой

$$\sum_{i=0}^{m} \phi_{\text{Пр}}(i).$$

Эта функция ϕ является рекурсивной.

Действительно, функция $\chi_0(m,n) = \phi_{\text{Пр}}(n)$ рекурсивна, так как есть

$$R([' [';\ \theta]], [\phi_{\text{Пр}};\ [';\ \eta_2^3]]).$$

Поэтому

$$\chi_1(m,l) = \sum_{i=0}^{l} \chi_0(m,i)$$

тоже рекурсивна. Однако ϕ есть $[\chi_1;\ \eta_1^1,\ \eta_1^1]$.

Пусть $\chi_2(n,m) = |(n+1) - \phi(m)|$. Тогда χ_2 — рекурсивная функция. Так как $p = M\chi_2$, то p — рекурсивная функция.

Пример 4.2.11.

Известно,что всякое натуральное число однозначно разлагается на простые множители. Через $ex(m,n)$ обозначим наибольшую степень числа $p(n)$, на которую натуральное число $m+1$ делится. Тогда $m+1$ не делится на $(p(n))^{ex(m,n)+1}$. Поэтому

$$ex = M([\overline{sg}; [rest; [';\ \eta_1^3], [\chi_3; [p;\ \eta_2^3], [';\ \eta_3^3]]]]),$$

где $\chi_3(x,y) = x^y$. Так как

$$\begin{aligned} \chi_3(x,0) =& 1 = [';\ \theta](x) \text{ и} \\ \chi_3(x,y+1) =& x^y * x = [*;\ \eta_3^3,\ \eta_1^3](x,y,\chi_3(x,y)), \end{aligned}$$

то $\chi_3 = R([';\ \theta], [*;\ \eta_1^3,\ \eta_3^3])$ и χ_3 — рекурсивная функция. Но тогда ex — тоже рекурсивная функция.

Пусть $exp(m,n) = ex(m \frown 1, n)$. Тогда exp — тоже рекурсивная функция. Если m отлично от 0, то $exp(m,n)$ — это наибольшая степень $p(n)$ на которую m делится. Если же $m = 0$, то $exp(m,n) = exp(0,n) = 0$ для любых n.

Например, $1050 = 105 * 10 = 7 * 15 * 10 = 2 * 3 * 5^2 * 7$. Поэтому $exp(1050,0) = 1$, $exp(1050,1) = 1$, $exp(1050,2) = 2$, $exp(1050,3) = 1$, $exp(1050,7) = 0$.

4.3 Программная вычислимость частично рекурсивных функций

Теорема 4.5. *Каждая частично рекурсивная функция программно вычислима.*

Доказательство. Индукцией по длине последовательности, задающей рассматриваемую частично рекурсивную функцию.

Базис индукции. Функция является простейшей. Программная вычислимость простейших функций установлена в примерах 4.1.2, 4.1.3, 4.1.4. Так как последняя функция либо простейшая, либо получается из предыдущих, которые по индукционному предположению программно вычислимы, суперпозицией, минимизацией или примитивной рекурсией, то индукционный шаг содержится в трех леммах.

Лемма 4.6. *Если функции ϕ, $\phi_1, \ldots, \phi_m$ программно вычислимы, то $[\phi;\ \phi_1, \ldots, \phi_m]$ тоже программно вычислима.*

Доказательство. Пусть Π, $\Pi_1, \ldots, \Pi_m$ вычисляют ϕ, $\phi_1, \ldots, \phi_m$ в x_1, а любые две различные из этих программ не имеют общих рабочих переменных. Пусть основные переменные Π есть $x_1, \ldots, x_m$, а основные переменные каждой из программ $\Pi_1, \ldots, \Pi_m$ есть $x_1, \ldots, x_n$.

В новой программе Р основными переменными будут $x_1, \ldots, x_n$, а остальные используемые переменные — рабочие. Пусть переменные

$$z_1, \ldots, z_n,\ y_1, \ldots, y_m$$

не используются в программах Π, $\Pi_1, \ldots, \Pi_m$.

Обозначим через P следующую программу:

$z_1 \leftarrow x_1; \ldots; z_n \leftarrow x_n;$

$\Pi_1;\ y_1 \leftarrow x_1; x_1 \leftarrow z_1; \ldots; x_n \leftarrow z_n;$

$\Pi_2;\ y_2 \leftarrow x_1; x_1 \leftarrow z_1; \ldots; x_n \leftarrow z_n;$

$\ldots;$

$\Pi_{m-1};\ y_{m-1} \leftarrow x_1; x_1 \leftarrow z_1; \ldots; x_n \leftarrow z_n;$

$\Pi_m;\ y_m \leftarrow x_1;$

$x_1 \leftarrow y_1; \ldots; x_m \leftarrow y_m;\ \Pi_n$

Ясно, что P вычисляет $[\phi;\ \phi_1, \ldots, \phi_m]$ в x_1. ∎

Лемма 4.7. *Если функции ϕ и χ программно вычислимы, то $R(\phi, \chi)$ тоже программно вычислима.*

Доказательство. Ради простоты обозначений, рассмотрим случай, когда ϕ одноместна, а χ трехместна. Пусть x — основная переменная программы Π_1, вычисляющей ϕ в x, а x, y, z — основные переменные программы Π_2, вычисляющей χ в x. Пусть рабочие переменные программы

Π_1 отличны от y и от рабочих переменных $z_1, \ldots, z_m$ программы Π_2. Пусть x_1, y_1, x_2 не используются программами Π_1 и Π_2. Рассмотрим следующую программу P:

$x_1 \leftarrow x;\ y_1 \leftarrow y;\ x_2 \leftarrow 0;\ \Pi_1;$
пока $x_2 < y_1$ делай
$z \leftarrow x;\ x \leftarrow x_1;\ y \leftarrow x_2;$
$\Pi_2; z_1 \leftarrow 0; \ldots; z_m \leftarrow 0;\ x_2 \leftarrow x_2'$
все

Ясно, что P вычисляет $R(\phi, \chi)$ в x, если x, y — основные переменные, а остальные используемые P переменные — рабочие.

В самом деле, пусть состояние σ всем рабочим переменным присваивает 0, $\sigma(x) = a$, $\sigma(y) = b$.

Пусть
$$\sigma_1 = (x_1 \leftarrow x;\ y_1 \leftarrow y;\ x_2 \leftarrow 0; \Pi_1)(\sigma).$$
Тогда $\sigma_1(x) = \phi(a)$, $\sigma_1(x_1) = a$, $\sigma_1(y_1) = b$, $\sigma_1(x_2) = 0$. Если $b = 0$, то $\sigma_1 = P(\sigma)$. Однако $R(\phi, \chi)(a, 0) = \phi(a)$. Так что $\phi_{\Pi,x}(a, 0) = R(\phi, \chi)(a, 0)$. Пусть $b > 0$. При выполнении тела цикла значение x_2 возрастает на 1, а значение y_1 не меняется. Поэтому тело цикла выполняется b раз. Пусть перед очередным выполнением тела цикла переменные x_1, x_2, x принимают значения a, i, $R(\phi, \chi)(a, i)$, а переменные $z_1, \ldots, z_m$ — значение 0. Тогда перед выполнением Π_2 рабочие переменные примут значение 0, а x, y, z — значения a, i, $R(\phi, \chi)(a, i)$. По определению Π_2, после выполнения
$$\Pi_2;\ z_1 \leftarrow 0; \ldots; z_m \leftarrow 0;\ x_2 \leftarrow x'$$
рабочие переменные Π_2 примут значение 0, значение x_1 не изменится и останется равным a, значение x_2 станет $i + 1$, а значение x будет равно $\chi(a, i, R(\phi, \chi)(a, i))$, но последнее, по определению $R(\phi, \chi)$, равно $R(\phi, \chi)(a, i + 1)$. Так как тело цикла выполняется b раз, а перед первым выполнением тела цикла x_1, x_2, x принимают значения a, 0, $R(\phi, \chi)(a, 0)$, то после выполнения цикла x_1, x_2, x примут значения a, b, $R(\phi, \chi)(a, b)$.

Поэтому, $\phi_{P,x}(a, b) = R(\phi, \chi)(a, b)$. ■

Лемма 4.8. *Если функция ϕ программно вычислима, то $(M\phi)$ тоже программно вычислима.*

Доказательство. Ради простоты обозначений, рассмотрим только случай, когда ϕ двухместна.

Пусть x, y — основные переменные программы П, вычисляющей ϕ в x. Пусть рабочие переменные программы П есть $z_1, \ldots, z_m$. Пусть x_1, y_1, x_2, x_3 не используются программой П.

Рассмотрим следующую программу P:

$x_1 \leftarrow x;\ x_2 \leftarrow 0;\ x_2 \leftarrow x';\ x_3 \leftarrow 0;\ y_1 \leftarrow 0;$

пока $y_1 < x_2$ делай

$x \leftarrow x_1;\ y \leftarrow x_3;\ \Pi; z_1 \leftarrow 0; \ldots; z_m \leftarrow 0;\ x_2 \leftarrow x;$

если $y_1 = x_2$ то

$x \leftarrow x_3$ иначе $x_3 \leftarrow x_3'$ конец

все

Ясно, что P вычисляет $(M\phi)$ в x, если x — основная переменная, а остальные используемые P переменные — рабочие.

В самом деле, пусть состояние σ всем рабочим переменным присваивает значение 0 и $\sigma(x) = a$. Тогда перед выполнением цикла x_1, x_2, x_3, y_1 примут значения a, 1, 0, 0. Поэтому тело цикла хотя бы один раз выполняется. Выполнение тела цикла не меняет значения y_1. Поэтому y_1 далее равно 0. Если перед следующим выполнением тела цикла значение x_2 больше 0, значения x_1, x_3 есть a, i, а значения $z_1, \ldots, z_m$ равны 0, то происходит еще одно выполнение тела цикла. После выполнения части тела цикла до условного оператора, по определению P, значения x_1, x_3 не меняются, значения $z_1, \ldots, z_m$ снова станут равны 0, а значение x_2 станет $\phi(a,i)$. Если $\phi(a,i) = 0$, то после выполнения тела цикла значение x_2 станет равным 0, значением x_3 останется i и выполнение цикла закончится. Но i в этом случае и есть $(M\phi)(a)$. Если же $\phi(a,i) > 0$, то после выполнения тела цикла значение x_2 будет больше 0, а значениями x_1, x_3 станут a, $i+1$, значения же $z_1, \ldots, z_m$ останутся равными 0. ∎

Доказательство теоремы 4.5 на этом заканчивается. ∎

4.4 Программы с метками

Определение 4.4.1. *Помеченным присваиванием называется выражение вида*

$$n : u \quad m$$

где n, m — натуральные числа, u — присваивание.

Число n называется меткой, а m — переходом.

Определение 4.4.2. *Помеченным тестом называется выражение вида*

$$n:\ t\ \ m_1\ m_2$$

где n, m_1, m_2 — натуральные числа, t — тест.

Число n называется меткой, m_1 называется положительным переходом, m_2 — отрицательным переходом.

Определение 4.4.3 (программы с метками). *Программой с метками называется конечная последовательность помеченных присваиваний и помеченных тестов, если различные члены этой последовательности имеют разные метки. Переменная используется программой с метками, если она используется одним из ее членов. Члены последовательности называются инструкциями.*

Определение 4.4.4 (работы программы с метками). *Работа программы с метками Π, не использующей переменных, отличных от $x_1, \ldots, x_m$, на состоянии σ описывается функциями ζ и δ, где ζ отображает ω в ω, а δ отображает $(\omega \times \{1, \ldots, m\})$ в ω (другими словами, δ каждой паре (a, b), где a — натуральное число, а b из $\{1, \ldots, m\}$, ставит в соответствие натуральное число). $\zeta(i)$ называется меткой инструкции, выполняемой в момент i, а $\delta(i, j)$ называется значением x_j в момент i.*

Функции ζ и δ определяются следующим образом.

$\zeta(0)$ — это метка первой инструкции, $\delta(0, j) = \sigma(x_j)$ для $j = 1, \ldots, m$.

Пусть $\zeta(i)$, $\delta(i, 1), \ldots, \delta(i, m)$ уже определены.

Если $\zeta(i)$ метит инструкцию

$$\zeta(i):\ u\ \ m$$

где u — присваивание, то $\zeta(i+1) = m$. Если u при этом имеет вид $x_k \leftarrow x'$, то $\delta(i+1, j) = \delta(i, j)$ для j, отличном от k, и $\delta(i+1, k) = \delta(i, k) + 1$. Если u имеет вид $x_k \leftarrow x_l$, то $\delta(i+1, j) = \delta(i, j)$ для j, отличном от k, и $\delta(i+1, k) = \delta(i, l)$. Если u имеет вид $x_k \leftarrow 0$, то $\delta(i+1, j) = \delta(i, j)$ для j, отличном от k, и $\delta(i+1, k) = 0$.

Если $\zeta(i)$ метит инструкцию

$$\zeta(i):\ t\ \ m_1\ m_2$$

где t — тест, то $\delta(i+1,j)=\delta(i,j)$ для $j=1,\dots,m$. Если t при этом имеет вид $x_k<x_l?$, то $\zeta(i+1)$ есть m_1, если $\delta(i,k)<\delta(i,l)$, и есть m_2 в остальных случаях. Если t при этом имеет вид $x_k=x_l?$, то $\zeta(i+1)$ есть m_1, если $\delta(i,k)=\delta(i,l)$, и есть m_2 в остальных случаях.

Если $\zeta(i)$ не метит никакую инструкцию, то $\zeta(i+1)=\zeta(i)$, $\delta(i+1,j)=\delta(i,j)$ для $j=1,\dots,m$. В этом случае говорим, что Π останавливается на σ в момент i. При этом, заключительным состоянием $\Pi(\sigma)$ называется такое состояние, что $\Pi(\sigma)(x_j)=\delta(i,j)$ для $j=1,\dots,m$ и $\Pi(\sigma)(y)=\sigma(y)$, если переменная y отлична от $x_1,\dots,x_m$.

Если Π не останавливается на σ ни в какой момент, то $\Pi(\sigma)$ не определено.

Если в определении 4.1.6 слово "структурированная"заменить на "с метками", получим определение функции, вычислимой программой с метками.

Определение 4.4.5. *Программа — это структурированная программа или программа с метками. Две программы Π_1 и Π_2 эквивалентны, если для каждого состояния σ либо оба состояния $\Pi_1(\sigma)$ и $\Pi_2(\sigma)$ не определены одновременно, либо $\Pi_1(\sigma)$ и $\Pi_2(\sigma)$ оба определены и $\Pi_1(\sigma)=\Pi_2(\sigma)$.*

Теорема 4.9. *Каждая структурированная программа эквивалентна некоторой программе с метками.*

Доказательство. Индукцией по суммарному числу композиций, условных операторов и циклов в программе. Если это число равно 0, то программа является присваиванием. Метим это присваивание числом 1, а в качестве перехода выберем число 2. Ясно, что программа с меткой, состоящая из этой одной получаемой инструкции, эквивалентна рассматриваемому присваиванию.

Пусть структурированная программа Π_1 эквивалентна программе с метками S_1 и Π_2 эквивалентна S_2. Можно считать,что в S_1 и S_2 нет общих меток. Переход в инструкции из S_1 на не существующую в S_1 метку заменим на переход на первую метку S_2, а переход в инструкции из S_2 на не существующую в S_2 метку заменим переходом на метку, не существующую ни в S_1, ни в S_2. Очевидно, что полученная последовательность инструкций, содержащая все переделанные инструкции как S_1, так и S_2, если первой взять переделанную первую инструкцию программы S_1, эквивалентна "$\Pi_1;\Pi_2$".

При тех же обозначениях для Π_1, Π_2, S_1, S_2 можно считать, что переходы в S_1 отличны от меток S_2, переходы в S_2 отличны от меток S_1, метки в S_1 не являются метками в S_2. Условный оператор "если ϕ то Π_1 иначе Π_2 конец"

заменяем тогда на программу с метками, первой инструкцией которой будет

$m: \ \phi? \ \ m_1 \ m_2$

где m не является ни меткой, ни переходом ни в S_1, ни в S_2, m_1 — метка первой инструкции S_1, m_2 — метка первой инструкции S_2, а затем идут все инструкции S_1 и S_2. Ясно, что полученная программа эквивалентна этому условному оператору. Цикл

"пока ϕ делай Π_1 все"

заменяем на программу с метками, первой инструкцией которой будет

$m: \ \phi? \ \ m_1 \ m_2$

где m не является ни меткой, ни переходом в S_1, m_2 отлично от всех меток S_1 и от m, m_1 — метка первой инструкции S_1, а затем идут все инструкции S_1, в которых все переходы на несуществующие метки заменены на m. Ясно, что полученная программа эквивалентна рассматриваемому циклу. ∎

Теорема 4.10. *Каждая функция, вычислимая программой с метками, также вычисляется некоторой структурированной программой вида*

"Π_1; пока $y = z$ делай Π_2 все ",

где Π_1 — композиция присваиваний, а Π_2 не содержит циклов.

Доказательство. Пусть рассматриваемая программа с метками Π содержит k инструкций. Можно считать, что они помечены числами $1, \ldots,$ k. Можно считать, что все переходы в Π лежат во множестве $\{1, \ldots, k,$ $k+1\}$. Пусть $y_1, \ldots, y_k$, z_1, z_2, y_{k+1} не используются в программе Π.

Каждую инструкцию вида

$n: \ u \ \ m$

где u — присваивание, заменим на структурированную программу:

"если $y_n = z_1$ то u; $y_n \leftarrow z_2$; $y_m \leftarrow z_1$ иначе $z_1 \leftarrow z_1$ конец "

Каждую инструкцию вида

$n: \ t \ \ m_1 \ m_2$

где t — тест вида $\phi?$, где ϕ — условие, заменим на структурированную программу:

"если $y_n = z_1$ то если ϕ то $y_n \leftarrow z_2$; $y_{m_1} \leftarrow z_1$

иначе $y_n \leftarrow z_2$; $y_{m_2} \leftarrow z_1$ конец иначе $z_1 \leftarrow z_1$ конец "

Пусть Π_2 — композиция всех вновь построенных структурированных программ, заменяющих инструкции программы Π.

Пусть Π_3 — это следующая структурированная программа:

"$z_1 \leftarrow 0$; $z_2 \leftarrow 0$; $z_2 \leftarrow z'$;

$y_1 \leftarrow z_1$; $y_2 \leftarrow z_2; \ldots; y_k \leftarrow z_2$; $y_{k+1} \leftarrow z_2$;

пока $y_{k+1} = z_2$ делай Π_2 все "

В качестве основных выберем те переменные, которые являются в Π основными. Пусть Π в x вычисляет некоторую функцию. Легко видеть, что Π_3 в x вычисляет ту же функцию. ∎

Таким образом, класс программно вычислимых функций не зависит от того, выбирается ли класс структурированных программ или класс программ с метками. Но из этих теорем можно получить и более глубокое следствие.

Теорема 4.11. *Если функция вычисляется некоторой программой, то она вычисляется и структурированной программой вида*

"Π_1; *пока* $y = z$ *делай* Π_2 *все* ",

где Π_1 *— это композиция присваиваний, а* Π_2 *не содержит циклов.*

Доказательство. Если функция вычисляется некоторой программой, то по теореме 4.9 она вычисляется программой с метками, а по теореме 4.10 — и программой указанного вида. ∎

4.5 Частичная рекурсивность программно вычислимых функций

Пусть Π — некоторая структурированная программа, которая не использует переменных, отличных от $x_1, \ldots, x_m$, а σ, σ_1 — состояния. Легко проверить и это уже использовалось, что если $\sigma_1(x_i) = \sigma(x_i)$ для $i = 1, \ldots, m$, то $\Pi(\sigma)$ и $\Pi(\sigma_1)$ одновременно определены или нет и $(\Pi(\sigma))(x_i) = (\Pi(\sigma_1))(x_i)$ для $i = 1, \ldots, m$. Другими словами, заключительные значения $x_1, \ldots, x_m$ являются функциями начальных значений $x_1, \ldots, x_m$ и не зависят от начальных значений остальных переменных. Обозначим эти функции через Y_{P,x_i} для $i = 1, \ldots, m$.

Теорема 4.12. *Какова бы ни была структурированная программа* Π, *указанного в теореме 4.10 вида, не использующая переменных, отличных от* $x_1, \ldots, x_m$, *все функции* $Y_{\Pi,x_1}, \ldots, Y_{\Pi,x_m}$ *частично рекурсивны.*

Доказательство. Это верно для случая, когда Π — присваивание. Если присваивание имеет вид $x_i \leftarrow x'$, $x_i \leftarrow 0$ или $x_i \leftarrow x_k$, то Y_{Π,x_j} есть η_j^m для j, отличного от i. В первом случае Y_{Π,x_i} есть $[';\ \eta_i^m]$, во втором есть $[\theta;\ \eta_i^m]$, в третьем есть η_k^m.

Если это верно для Π_1 и Π_2, то верно и для $\Pi_1;\ \Pi_2$, так как

$$Y_{\Pi_1;\Pi_2,\ x_i} = [Y_{\Pi_2,\ x_i};\ Y_{\Pi_1,\ x_1}, \ldots, Y_{\Pi_1,\ x_m}].$$

Пусть Π не содержит циклов и имеет вид

"если $x_k = x_l$ то Π_1 иначе Π_2 конец ".

Так как Π_1 и Π_2 не содержат циклов, то функции $Y_{\Pi_1,\ x_i}$ и $Y_{\Pi_2,\ x_i}$ всюду определенные. Если они частично рекурсивны, то они рекурсивны. Ясно, что $Y_{\Pi,\ x_i}$ есть $Y_{\Pi_1,\ x_i}$, если начальные значения x_k и x_l совпадают, и есть $Y_{\Pi_2,\ x_i}$, если начальные значения x_l и x_k различны. Поэтому Y_{Π,x_i} получается из $Y_{\Pi_1,\ x_i}, Y_{\Pi_2,\ x_i}, [|-|;\ \eta_k^m;\ \eta_l^m]$ разбором случаев, где $|-|$ обозначает модуль разности. По теореме 4.4, $Y_{\Pi,\ x_i}$ тоже рекурсивна.

Случай, когда Π имеет вид

"если $x_k < x_l$ то Π_1 иначе Π_2 конец",

а Π_1 и Π_2 не содержат циклов, рассматривается аналогично. Надо только $[|-|;\ \eta_k^m;\ \eta_l^m]$ заменить на $[\frown;\ [';\ \eta_k^m],\ \eta_l^m]$.

Таким образом, теорема верна для Π, не содержащей циклов.

Пусть Π имеет вид

"пока $x_k = x_l$ делай Π_1 все"

и для Π_1 теорема верна, где Π_1 не содержит циклов. Тогда $Y_{P_1,\ x_1}, \ldots, Y_{P_1,\ x_m}$ являются рекурсивными функциями.

Пусть

$$b_1 = Y_{\Pi_1,\ x_1}(a_1, \ldots, a_m), \ldots, b_m = Y_{P_1,\ x_m}(a_1, \ldots, a_m).$$

Рассмотрим функцию Y, определяемую так, что

$$Y(a_1, \ldots, a_m) = 3^{b_1} * \cdots * p(m)^{b_m}.$$

Пусть $\Xi_0 = [\theta;\ \eta_1^1]$ и $\Xi_{i+1} = [';\ \Xi_i]$. Все функции $\Xi_0, \ldots, \Xi_m$ рекурсивны. Пусть J — это возвышение в степень: $J(a,b) = a^b$. Рекурсивность J отмечена в примере 4.2.11. Пусть

$$B_1 = [J;\ [p;\ \Xi_1], Y_{\Pi_1,\ x_1}]$$

и пусть

$$B_{i+1} = [*;\ B_i,\ [J;\ [p;\ \Xi_{i+1}], Y_{P_1,\ x_{i+1}}]]$$

для $i = 1, \ldots, m-1$. Ясно, что все функции $B_1, \ldots, B_m$ рекурсивны, но Y есть B_m. Поэтому Y есть рекурсивная функция.

Пусть $\phi(x, 0) = x$ и

$$\phi(x, y+1) = Y(exp(\phi(x,y), 1), \ldots, exp(\phi(x,y), m)).$$

Так как

$$exp(\phi(x,y), i) = [exp;\ \eta_3^3,\ s_i](x, y, \phi(x,y)),$$

где $s_0 = [\theta;\ \eta_1^3]$ и $s_{i+1} = [';\ s_i]$, и

$$\phi(x, y+1) = [Y;\ [exp;\ \eta_3^3,\ s_1], \ldots, [exp;\ \eta_3^3,\ s_m]](x, y, \phi(x, y)),$$

то

$$\phi = R(\eta_1^1, [Y;\ [exp;\ \eta_3^3,\ s_1], \ldots, [exp;\ \eta_3^3,\ s_m]])$$

и ϕ рекурсивна.

Менее формально, набор значений $a_1, \ldots, a_m$ переменных $x_1, \ldots, x_m$ полностью определяется произведением $3^{a_1} * \cdots * p(m)^{a_m}$ и задается этим произведением a. После выполнения программы Π_1 новые значения $x_1, \ldots, x_m$ задаются числами

$$Y_{\Pi_1,\ x_1}(a_1, \ldots, a_m), \ldots, Y_{\Pi_1,\ x_m}(a_1, \ldots, a_m).$$

Новый набор значений задается числом $Y(a_1, \ldots, a_m)$. Так как $a_i = exp(a, i)$, то

$$Y(a_1, \ldots, a_m) = Y(exp(a, 1), \ldots, exp(a, m)) = \phi(a, 1).$$

Число $\phi(a, i)$ задает набор значений переменных, который получится после i выполнений программы Π_1.

Пусть

$$\phi_1(a_1, \ldots, a_m) = 3^{a_1} * \cdots * p(m)^{a_m}.$$

Пусть

$$J_1 = [J;\ [p;\ \Xi_1];\ \eta_1^m]$$

и

$$J_{i+1} = [*;\ J_i,\ [J;\ [p;\ \Xi_{i+1}],\ \eta_{i+1}^m]]$$

для $i = 1, \ldots, m-1$. Ясно, что функции $J_1, \ldots, J_m$ рекурсивны, но ϕ_1 есть J_m. Поэтому ϕ_1 — рекурсивная функция.

Пусть

$$\phi_2 = [\phi;\ [\phi_1;\ \eta_1^{m+1}, \ldots, \eta_m^{m+1}],\ \eta_{m+1}^{m+1}].$$

Ясно, что ϕ_2 — рекурсивная функция и

$$\phi_2(a_1, \ldots, a_m, b) = \phi(\phi_1(a_1, \ldots, a_m), b).$$

Наконец, пусть

$$Y_i(a_1, \ldots, a_m, b) = exp(\phi_2(a_1, \ldots, a_m, b), i)$$

для $i = 1, \ldots, m$. Если $D_0 = [\theta; \eta_1^{m+1}]$ и $D_{i+1} = ['; \ D_i]$, то $Y_i = [exp; \ \phi_2, \ D_i]$ и, значит, Y_i —- рекурсивная функция. Ясно, что

$$Y_i(a_1, \ldots, a_m, b)$$

является значением x_i после b выполнений программы Π_1. Пусть

$$Y_0 = [\overline{sg}; \ [| - |; \ Y_k, \ Y_l]].$$

Тогда $(MY_0)(a_1, \ldots, a_m)$ есть наименьшее из таких b, что $Y_k(a_1, \ldots, a_m, b)$ отлично от $Y_l(a_1, \ldots, a_m, b)$. Понятно,что

$$Y_{\Pi, \ x_i} = [Y_i; \ \eta_1^m, \ldots, \eta_m^m, \ MY_0].$$

Это доказывает частичную рекурсивность $Y_{\Pi, \ x_i}$.

В общем случае Π имеет вид композиции структурированной программы, не содержащей циклов, и цикла указанного вида. Как отмечалось в начале доказательства, если теорема верна для двух программ, то она верна и для их композиции. ∎

Теорема 4.13. *Каждая программно вычислимая функция является частично рекурсивной.*

Доказательство. Каждая программно вычислимая функция для некоторой структурированной программы Π указанного в теореме 4.10 вида получается из некоторой $Y_{\Pi, \ x}$ подстановкой нулей вместо некоторых аргументов. Такая подстановка, конечно, не выводит из класса частично рекурсивных функций. ∎

4.6 Неразрешимые проблемы

Занумеруем сначала все переменные натуральными числами так, чтобы каждая переменная имела один номер и разные переменные имели разные номера. При этом можно считать, что каждое натуральное число является номером некоторой переменной. В этом параграфе через x_i обозначаем переменную с номером i.

Номером присваивания

$$x_i \leftarrow 0$$

называем число $2 * 3^i$, номером

$$x_i \leftarrow x_j$$

называем число $2^2 * 3^i * 5^j$ и номером

$$x_i \leftarrow x_i'$$

называем число $2^3 * 3^i$.

Номером композиции программ Π_1 и Π_2 с номерами n и m называем $2^4 * 3^n * 5^m$, номером условия $x_i = x_j$ называем $2^5 * 3^i * 5^j$, номером условия $x_i < x_j$ называем $2^6 * 3^i * 5^j$, номером условного оператора

"если Φ то Π_1 иначе Π_2 конец"

называем число $2^7 * 3^n * 5^m * 7^l$, где n задает номер условия Φ, m задает номер Π_1, l задает номер Π_1. Номером цикла

"пока Φ делай Π_1 все"

называем число $2^8 * 3^n * 5^m$, где n задает номер условия Φ, m задает номер Π_1.

Теорема 4.14. *Никакое натуральное число не является номером двух разных структурированных программ. Каждая структурированная программа имеет хотя бы один номер.*

Доказательство. Второе утверждение очевидно. Первое доказываем индукцией.

Пусть n является номером программ Π_1 и Π_2. Пусть $m = exp(n, 0)$. Если $exp(n, 0) = 1$, то Π_1 и Π_2 являются присваиваниями

$$x_{exp(n,1)} \leftarrow 0;$$

если $exp(n, 0) = 2$, то Π_1 и Π_2 являются присваиваниями

$$x_{exp(n,1)} \leftarrow x_{exp(n,2)};$$

если $exp(n, 0) = 3$, то Π_1 и Π_2 являются присваиваниями

$$x_{exp(n,1)} \leftarrow x'_{exp(n,1)}.$$

Если $exp(n, 0) = 4$, то Π_1 есть композиция

$$"\Pi_{1,1};\ \Pi_{1,2}"$$

и Π_2 есть композиция

$$"\Pi_{2,1};\ \Pi_{2,2}".$$

Так как номера $\Pi_{1,1}$ и $\Pi_{2,1}$ равны $exp(n, 1)$, а номера $\Pi_{1,2}$ и $\Pi_{2,2}$ равны $exp(n, 2)$, то по индукционному предположению $\Pi_{1,1}$ совпадает с $\Pi_{2,1}$, а $\Pi_{1,2}$ совпадает с $\Pi_{2,2}$.

Если $exp(n, 0) = 7$, то Π_1 имеет вид

"если Φ_1 то $\Pi_{1,1}$ иначе $\Pi_{1,2}$ конец ",

где $exp(n, 1)$ задает номер Φ_1, $exp(n, 2)$ задает номер $\Pi_{1,1}$, $exp(n, 3)$ задает номер $\Pi_{1,2}$. При этом Π_2 имеет вид

"если Φ_2 то $\Pi_{2,1}$ иначе $\Pi_{2,2}$ конец ",

где $exp(n, 1)$ задает номер Φ_2, $exp(n, 2)$ задает номер $\Pi_{2,1}$, $exp(n, 3)$ задает номер $\Pi_{2,2}$. По индукционному предположению, $\Pi_{1,1}$ совпадает с $\Pi_{2,1}$, $\Pi_{1,2}$ совпадает с $\Pi_{2,2}$. Поэтому Π_1 совпадает с Π_2.

Если $exp(n, 0) = 8$, то Π_1 имеет вид

"пока Φ_1 делай $\Pi_{1,1}$ все ",

а Π_2^m имеет вид

"пока Φ_2 делай $\Pi_{2,1}$ все ".

Так как $exp(n, 1)$ является номером Φ_1 и Φ_2, то Φ_1 совпадает с Φ_2. Так как $exp(n, 2)$ является номером $\Pi_{1,1}$ и $\Pi_{2,1}$, то по индукционному предположению $\Pi_{1,1}$ совпадает с $\Pi_{2,1}$. Поэтому Π_1 совпадает с Π_2. ∎

Через v обозначим отображение, которое Π ставит в соответствие множество всех номеров структурированной программы Π.

В определении 4.1.6 функции, вычисляемой программой, фиксируется некоторая переменная x. Договоримся в качестве x выбрать x_0.

Пусть структурированная программа Π имеет номер n. Тогда одним из номеров программы

$$\underbrace{x_0 \leftarrow x_0'; \ldots; x_0 \leftarrow x_0';}_{n \text{ раз}}\ \Pi \tag{13}$$

будет следующее число. Пусть $\phi(a, 0) = a$ и $\phi(a, b+1) = 2^4 * 3^{\phi(a,b)} * 5^a$.

Так как 8 — это номер присваивания $x_0 \leftarrow x_0'$, то $\phi(8, i)$ задает номер композиции

$$\underbrace{x_0 \leftarrow x_0'; \ldots; x_0 \leftarrow x_0';}_{(i+1) \text{ раз}}$$

Пусть $\Xi_0 = [\theta;\ \eta_1^3]$; $\Xi_{i+1} = [';\ \Xi_i]$. Все функции Ξ_i рекурсивны. Так как

$$\phi = R(\eta_1^1,\ [*;\ \Xi_{16},\ [*;\ [J;\ \Xi_3,\ \eta_3^3],\ [J;\ \Xi_5,\ \eta_1^3]]]),$$

где $J(a, b) = a^b$, то ϕ — рекурсивная функция. Пусть

$$\phi_1 = [\phi;\ [\Xi_8;\ \eta_1^1,\ \eta_1^1,\ \eta_1^1],\ \eta_1^1].$$

Тогда ϕ_1 тоже рекурсивна. Как уже отмечалось, $\phi_1(i)$ это номер композиции

$$\underbrace{x_0 \leftarrow x_0'; \ldots ; x_0 \leftarrow x_0';}_{(i+1) \text{ раз}}$$

Теперь

$$Y(n) = 2^4 * 3^{\phi_1(n \dot{-} 1)} * 5^n$$

задает номер программы (13). Тогда

$$Y = [*;\ [\Xi_{16};\ \eta_1^1,\ \eta_1^1,\ \eta_1^1], \\ [*;\ [J;\ [\Xi_3;\ \eta_1^1,\ \eta_1^1,\ \eta_1^1],\ [\phi_1; (-\ :)]],\ [J;\ [\Xi_5;\ \eta_1^1,\ \eta_1^1,\ \eta_1^1],\ \eta_1^1]]].$$

Поэтому Y есть рекурсивная функция. Значит, Y программно вычислима (теорема 4.5).

Итак, нами доказана

Теорема 4.15. *Отображение v является правильной нумерацией структурированных программ.*

Теорема 4.16. *Не существует алгоритма, определяющего по паре натуральных чисел, являются ли эти числа номерами эквивалентных структурированных программ в нумерации v.*

Доказательство. От противного. Пусть такой алгоритм существует. Тогда программно вычислима и, по теореме 4.13, частично рекурсивна функция ϕ, равная 1, если n и m являются номерами эквивалентных структурированных программ, и равная 0 в остальных случаях. Легко заметить, что если m есть номер структурированной программы Π, а переменная x_i используется этой программой, то $i < m$. Один из номеров программы

"$x_0 \leftarrow 0; \ldots ; x_i \leftarrow 0$"

есть число $s(i)$, определяемое следующим образом: $s(0) = 2$ и

$$s(i+1) = 2^4 * 3^i * 5^{2*3^{i+1}}.$$

В качестве упражнения предлагаем читателю доказать рекурсивность s. Для этого рассмотреть сначала функцию s_1 такую, что $s_1(x,y) = s(y)$ и доказать, что s_1 рекурсивна. Осталось заметить, что $s = [s_1;\ \eta_1^1,\ \eta_1^1]$.

Поэтому, если

$$Y(n) = 2^4 * 3^{s(n)} * 5^{Y_1(n)},$$

где
$$Y(n) = 2^4 * 3^n * 5^{s(n)},$$
то $Y(n)$ является одним из номеров программы
$$x_0 \leftarrow 0; \ldots ; x_n \leftarrow 0;\ \Pi; x_0 \leftarrow 0; \ldots ; x_n \leftarrow 0,$$
если n является одним из номеров программы П. При этом функция Y является рекурсивной. Эта программа останавливается на нуле тогда и только тогда, когда П останавливается на нуле, и эта программа эквивалентна программе
$$x_0 \leftarrow 0; \ldots ; x_n \leftarrow 0$$
тогда и только тогда, когда П останавливается на 0. Число $s(n)$ является одним из номеров последней программы. Значит, n тогда и только тогда является номером структурированной программы, останавливающейся на 0, когда $\phi(Y(n), s(n)) = 1$. Если положить $\phi_1 = [\phi;\ Y,\ s]$, то ϕ_1 является рекурсивной функцией и, по теореме 4.5, программно вычислима. Кроме того, $\phi_1(n) = 1$, если n — номер структурированной программы, останавливающейся на 0, и $\phi_1(n) = 0$ в остальных случаях. Это противоречит теореме 4.2. ∎

В теореме 3 вместо "эквивалентных"можно написать "вычисляющих одну и ту же функцию в x_0", имея в виду, что x_0 — основная переменная, а остальные используемые переменные — рабочие. В доказательстве надо заменить определение ϕ, считая $\phi(n,m) = 1$, если n и m — номера программ, вычисляющих одну и ту же функцию, и $\phi(n,m) = 0$ в остальных случаях, и заметить, что рассматриваемые программы тогда и только тогда вычисляют в x_0 одну и ту же функцию, когда П останавливается на 0. Таким образом, нами доказана

Теорема 4.17. *Не существует алгоритма, определяющего по паре натуральных чисел, являются ли эти числа номерами структурированных программ, вычисляющих одну и ту же функцию в* x_0.

Упражнения к §§4.1—4.6

Упражнение 4.6.1. *Построить программы и доказать их корректность для вычисления следующих функций:*

а) $3 * (x!)$;

б) $2 + (x!)$;

в) целая часть кубического корня плюс один;

г) целая часть квадратного корня минус три, если это число неотрицательно, или нуль;

д) наибольшее из квадрата первого числа и корня квадратного из второго;

е) наименьшее из двух чисел;

ж) равной $x!$, *если* $x < y$, *и* $y!$ *в остальных случаях;*

з) равной 0, если $x < y$, *и* 1, *если* $x \geq y$;

и) равной $|x - y|$, *если* x *четно, и* $2 * x$ *в остальных случаях;*

к) равной 0, если $x > 5$, *и* y *в остальных случаях;*

л) равной разности x *и* y, *если* $x > y$ *и* x *— простое число; нулю, если* $x \leq y$ *и* x *— простое число;* x *в остальных случаях;*

м) целая часть корня седьмой степени из произведения x *и* y;

н) равной x, *если* y *— простое число, и 2 в остальных случаях;*

о) $R(\eta_1^1, [*;\ \eta_3^3,\ \eta_1^3])$;

п) $R(\eta_1^1, [+;\ \eta_3^3,\ \eta_2^3])$;

р) $R(\eta_1^2, [*;\ \eta_3^4,\ \eta_4^4])$.

Упражнение 4.6.2. *Доказать рекурсивность следующих функций:*

а) $3 * (x!)$;

б) $2 + (x!)$;

в) целая часть кубического корня плюс один;

г) целая часть квадратного корня минус три, если это число неотрицательно, или нуль;

д) наибольшее из квадрата первого числа и корня квадратного из второго;

е) наименьшее из двух чисел;

ж) равной $x!$, *если* $x < y$, *и* $y!$ *в остальных случаях;*

з) равной 0, *если* $x < y$, *и 1, если* $x \geq y$;

и) равной $|x - y|$, *если* x *четно, и* $2 * x$ *в остальных случаях;*

к) равной x, *если* $x > 5$, *и* y *в остальных случаях;*

л) равной разности x *и* y, *если* $x > y$ *и* x *— простое число, нулю, если* $x \leq y$ *и* x *— простое число,* x *в остальных случаях;*

м) целая часть корня седьмой степени из произведения x *и* y;

н) равной x, *если* y *— простое число, и 2 в остальных случаях.*

Упражнение 4.6.3. *Доказать, что функции, вычисляемые следующими программами, являются частично рекурсивными и привести их явные задания в виде рекурсивных термов (как, например, в упражнении 4.6.1):*

а)

$i \leftarrow 0$; *если* $x = i$ *то* $x \leftarrow x$ *иначе если* $y = i$ *то* $x \leftarrow x$ *иначе пока* $x < y$ *делай* $y \leftarrow y$; $i \leftarrow i'$; $x \leftarrow x'$ *все конец конец*

б)

$i \leftarrow 0$; *пока* $x < y$ *делай* $i \leftarrow i'$; $x \leftarrow x'$ *все*

в)

$t \leftarrow 0$; $i \leftarrow 0$; *пока* $i < x$ *делай* $s \leftarrow y$; $j \leftarrow 0$; *пока* $j < y$ *делай* $t \leftarrow t'$; $t \leftarrow t'$; $j \leftarrow j'$ *все;* $i \leftarrow i'$; $i \leftarrow i'$ *все*

г)

$r \leftarrow 0$; $q \leftarrow x$; *пока* $y < q$ *делай* $i \leftarrow 0$; $j \leftarrow y$; *пока* $j < q$ *делай* $i \leftarrow i'$; $j \leftarrow j'$ *все;* $q \leftarrow i$; $r \leftarrow r'$ *все*

Упражнение 4.6.4. *Построить программы с метками, эквивалентные программам из упражнения 4.6.3.*

Упражнение 4.6.5. *Построить структурированные программы без вложенных циклов, эквивалентные программам из упражнений 4.6.3.в) и 4.6.3.г).*

4.7 Машины Тьюринга

Рассмотрим алфавит $E = \{\wedge, |, *, 1, 2, 3\}$. Символ $\wedge$ называется пустым. Через е будем в этом и следующих параграфах этой главы обозначать любой из шести символов этого алфавита, а через d — любой из символов $\{\wedge, *\}$. E назовем внешним алфавитом, а его элементы — обозреваемыми символами. Заметим, что e **в дальнейшем не обозначает пустое слово**.

Пусть Q — некоторый другой конечный алфавит такой, что Q и E не имеют общих элементов.

Элементы Q назовем состояниями, а Q — внутренним алфавитом или алфавитом состояний. Символы $!, h$ не входят ни в E, ни в Q. Символ ! называется заключительным состоянием. В Q выделяется один элемент, называемый начальным состоянием.

Определение 4.7.1 (машины Тьюринга). *Машиной Тьюринга (МТ) с множеством состояний Q и начальным состоянием q из Q называется отображение, которое каждой паре, составленной из обозреваемого символа и состояния из Q, ставит в соответствие тройку, первый элемент которой задает новый обозреваемый символ, второй — один из символов R, L, C, третий — новое состояние из Q или заключительное состояние.*

Таким образом, если множество всех троек, первый элемент которых взят из E, второй — из $\{R, L, C\}$, третий — из Q или $\{!\}$, обозначить через

$$E \times \{R, L, C\} \times (Q \bigcup \{!\}),$$

то МТ M — это совокупность множества состояний Q, начального состояния q из Q и отображения $(E \times Q)$ в $(E \times \{R, L, C\} \times (Q \bigcup \{!\}))$, где $(E \times Q)$ — это множество всех пар, первый элемент которых взят из E, а второй из Q.

МТ M изображается таблицей, имеющей 6 столбцов, помеченных элементами E, и столько строк, сколько элементов в Q. Эти строки метятся элементами Q. На пересечении столбца с меткой e и строки с меткой q стоит тройка $M(e, q)$, которая ставится в соответствие паре $\langle e, q\rangle$. Договоримся, что первая строка метится начальным состоянием.

Задавая МТ таблицей, мы часто не будем заполнять некоторые клетки, если тройки, которые должны стоять в незаполненных клетках, не будут использоваться в рассуждениях. В тройке не пишем C, не пишем новое состояние, если оно совпадает со старым, не пишем новый обозреваемый символ, если он совпадает со старым. Если D — некоторый алфавит, то слово в алфавите D — это либо выражение вида Aa, где A — вещь, которая является словом в алфавите D, а a входит в D, либо пустое слово.

Через D^* обозначают множество всех слов в алфавите D.

Определение 4.7.2. *Словом Поста (СП) для МТ M с множеством состояний Q и начальным состоянием q из Q называется выражение вида $hArsBh$, где A и B — слова в алфавите E, r из $Q \bigcup \{!\}$, s из E. Одно из слов A и B или оба они могут быть пустыми. Однако слово A не начинается на $\wedge$ и слово B не оканчивается на $\wedge$. Если r — начальное состояние, то слово Поста называется начальным, а если r — заключительное состояние, то заключительным.*

МТ перерабатывает свое слово Поста в новое слово Поста, кроме случая, когда рассматриваемое слово Поста — заключительное.

Определение 4.7.3 (работы МТ на СП). *Пусть M — МТ с множеством состояний Q и Π — СП для M.*

Если Π — заключительное СП, то $M(\Pi)$ есть Π.

Пусть Π есть $hArsBh$ и r из Q.

Случай 1. $M(s, r) = s'Lr'$, s' — непустой символ, A есть A_1a, a из E, A_1 из E^*. Тогда $M(\Pi)$ есть $hA_1r'as'Bh$.

Случай 2. $M(s,r) = s'Lr'$, s' *— непустой символ,* A *- пустое слово. Тогда* $M(\Pi)$ *есть* $hr' \wedge s'Bh$.

Случай 3. $M(s,r) = \wedge Lr'$, A *есть* A_1a, a *из* E, A_1 *из* E^*, B *— непустое слово. Тогда* $M(\Pi)$ *есть* $hA_1r'a \wedge Bh$.

Случай 4. $M(s,r) = \wedge Lr'$, A *есть* A_1a, a *из* E, A_1 *из* E^*, B *— пустое слово. Тогда* $M(\Pi)$ *есть* $hA_1r'ah$.

Случай 5. $M(s,r) = \wedge Lr'$, A *— пустое слово,* B *— непустое слово. Тогда* $M(\Pi)$ *есть* $hr' \wedge \wedge Bh$.

Случай 6. $M(s,r) = \wedge Lr'$, A *и* B *— пустые слова. Тогда* $M(\Pi)$ *есть* $hr' \wedge h$.

Случай 7. $M(s,r) = s'Rr'$, s' *— непустой символ,* B *есть* bB_1, b *из* E, B_1 *из* E^*. *Тогда* $M(\Pi)$ *есть* $hAs'r'bB_1h$.

Случай 8. $M(s,r) = s'Rr'$, s' *— непустой символ,* B *— пустое слово. Тогда* $M(\Pi)$ *есть* $hAs'r' \wedge h$.

Случай 9. $M(s,r) = \wedge Rr'$, B *есть* bB_1, b *из* E, B_1 *из* E^*, A *— непустое слово. Тогда* $M(\Pi)$ *есть* $hA \wedge r'bB_1h$.

Случай 10. $M(s,r) = \wedge Rr'$, B *есть* bB_1, b *из* E, B_1 *из* E^*, A *— пустое слово. Тогда* $M(\Pi)$ *есть* $hr'bB_1h$.

Случай 11. $M(s,r) = \wedge Rr'$, B *— пустое слово,* A *— непустое слово. Тогда* $M(\Pi)$ *есть* $hA \wedge r' \wedge h$.

Случай 12. $M(s,r) = \wedge Rr'$, A *и* B *— пустые слова. Тогда* $M(\Pi)$ *есть* $hr' \wedge h$.

Случай 13. $M(s,r) = s'Cr'$. *Тогда* $M(\Pi)$ *есть* $hAr's'Bh$.

Говорят, что МТ M *за один такт работы перерабатывает СП* Π *в СП* $M(\Pi)$, *и пишут:* $\Pi \Longrightarrow^1_M M(\Pi)$. *Если* $\Pi \Longrightarrow^1_M \Pi_1$ *и* $\Pi_1 \Longrightarrow^n_M \Pi_2$, *то* $\Pi \Longrightarrow^{n+1}_M \Pi_2$. *Запись* $\Pi \Longrightarrow^+_M \Pi_1$ *означает, что существует такое положительное натуральное* i, *что* $\Pi \Longrightarrow^i_M \Pi_1$. *В этом случае говорят, что* M *перерабатывает* Π *в* Π_1. *Запись* $\Pi \Longrightarrow^*_M \Pi_1$ *означает, что либо* $\Pi \Longrightarrow^+_M \Pi_1$, *либо* Π_1 *есть* Π.

Менее формально R — это движение на один символ вправо, L — влево, C — отсутствие движения. В слове Поста $hArsBh$ состояние расположено перед обозреваемым символом.

Сначала происходит замена обозреваемого символа на новый, потом движение вправо, если указано R, или влево, если указано L, на один символ. В новом СП отбрасывается первый или последний пустой символ, но обозреваемый символ всегда сохраняется. Если обозревается последний символ и движение вправо или если обозревается первый символ и движение влево, то добавляется пустой символ в качестве нового обозреваемого, соответственно, справа или слева.

Пример 4.7.1.

Рассмотрим МТ Пра$_e$ при $e = *$.

Пра$_e$	$\wedge$	$\mid$	$*$	1	2	3
q	Rq_1	Rq_1	Rq_1	Rq_1	Rq_1	Rq_1
q_1	R	R	!	R	R	R

Пра$_e$ имеет две строки. Во второй строке во всех столбцах, кроме помеченного e, стоит R, а в столбце, помеченном e, стоит !. В первой строке во всех столбцах стоит Rq_1. Имеется два состояния: начальное q и q_1. Рассмотрим СП $hq\wedge|*1||*h$ для Пра$_*$. Так как Пра$_*(\wedge, q)$ есть $\wedge Rq_1$, то имеет место случай 10, где B есть $|*1||*$. Поэтому Пра$_*$ указанное слово Π_0 перерабатывает в слово Π_1 =Пра$_*(\Pi_0)$, равное $hq_1|*1||*h$. Так как Пра$_*(|, q_1)$ есть $|Rq_1$, то Π_1 перерабатывается в Π_2, равное $h|q_1*1||*h$ (случай 7 при пустом A и B, равном $*1||*$). Так как Пра$_*(*, q_1)$ есть $*C!$, то Π_2 перерабатывается в $h|!*1||*h$. Итак,

$$hq\wedge|*1||*h \Longrightarrow^3_{\text{Пра}_*} h|!*1||*h.$$

Слово Поста $h|!*1||*h$ является заключительным.

Определение 4.7.4 (ленты МТ и конфигурации). *Лентой называется отображение δ начального отрезка $\{0, 1, \dots, n\}$ натуральных чисел в E. Говорят, что на ленте δ в ячейке i записан символ $\delta(i)$. n называется концом ленты δ, если δ отображает $\{0, 1, \dots, n\}$ в E.*

Конфигурацией МТ M называется тройка $\langle q, i, \delta\rangle$, где q — состояние M или заключительное состояние, δ — лента, i — натуральное число, а $\delta(i)$ определено.

Определение 4.7.5 (работы МТ на ленте). *Пусть M - МТ с множеством состояний Q, $\langle q, i, \delta\rangle$ — конфигурация M. Если q — заключительное, то*

$$M(q, i, \delta) = \langle q, i, \delta\rangle.$$

Пусть q входит в Q.

Случай 1. $M(\delta(i), q) = sLr$, i отлично от 0. Тогда

$$M(q, i, \delta) = \langle r, i-1, \delta'\rangle,$$

где δ' — такая лента, что $\delta'(j) = \delta(j)$ для всех таких отличных от i натуральных чисел, которые не превосходят конца ленты δ, $\delta'(i) = s$, концы δ и δ' совпадают.

Случай 2. $M(\delta(i),q) = sLr$, $i = 0$. *В этом случае* $M(q,i,\delta)$ *не определено. МТ* M *сходит слева с ленты.*

Случай 3. $M(\delta(i),q) = sRr$, i *меньше конца ленты* δ. *Тогда*

$$M(q,i,\delta) = \langle r, i+1, \delta' \rangle.$$

Случай 4. $M(\delta(i),q) = sRr$, i *— конец ленты* δ. *В этом случае МТ* M *сходит с ленты справа, а* $M(q,i,\delta)$ *не определено.*

Случай 5. $M(\delta(i),q) = sCr$. *В этом случае*

$$M(q,i,\delta) = \langle r, i, \delta' \rangle.$$

Говорят, что МТ M *перерабатывает конфигурацию*

$$\langle q, i, \delta \rangle$$

в

$$M(q,i,\delta).$$

Лента изображается последовательностью $\delta(0)\ldots\delta(n)$, $\langle q,i,\delta\rangle$ изображается как

$$\delta(0)\ldots\delta(i-1)q\delta(i)\ldots\delta(n)$$

при $i > 0$ и как

$$q\delta(0)\ldots\delta(n)$$

при $i = 0$. При этом e^0 изображает пустое слово, а e^{i+1} изображает $e^i e$.

Например, $|^3$ изображает $|||$ и $*q|||*1*$ может изображаться как $*q|^3*1*$, а $q**1*$ как $q*|^0*1*$.

Если A есть $a_1\ldots a_s$, где $a_1,\ldots,a_s$ из E,

$$K = \delta(0)\ldots\delta(i-1)q\delta(i)\ldots\delta(n),$$

то AK есть

$$a_1\ldots a_s\delta(0)\ldots\delta(i-1)q\delta(i)\ldots\delta(n)$$

и KA есть

$$\delta(0)\ldots\delta(i-1)q\delta(i)\ldots\delta(n)a_1\ldots a_s.$$

Менее формально, i называют обозреваемой ячейкой конфигурации $\langle q,i,\delta\rangle$. МТ M меняет содержимое обозреваемой ячейки на первый элемент тройки $M(\delta(i),q)$, сдвигается на одну ячейку влево, вправо или вообще не сдвигается в зависимости от того, равен второй элемент указанной тройки L, R или C, и переходит в новое состояние, указанное третьим элементом указанной тройки.

Определение 4.7.6. *Пусть K, K' — конфигурации МТ M и $M(K) = K'$. Тогда пишем: $K \longmapsto^1_M K'$. Пишем $K \longmapsto^{i+1}_M K'$, если существует такая конфигурация K_1 МТ M, что $K \longmapsto^i_M K_1$ и $K_1 \longmapsto^1_M K'$. Пишем $K \longmapsto^+_M K'$, если существует такое $i > 0$, что $K \longmapsto^i_M K'$. Пишем $M : K \longmapsto K'$, если K' имеет вид $\langle !, i, \delta \rangle$ и $K \longmapsto^+_M K'$.*

Скажем, что конфигурация МТ $M \langle q, i, \delta_1 \rangle$ продолжает конфигурацию $\langle q, i, \delta \rangle$, если конец n_1 ленты δ_1 не меньше конца n ленты δ, $\delta_1(i) = \wedge$ для $i > n$, и $\delta_1(i) = \delta(i)$ для $i = 0, 1, \ldots, n$.

Пишем $M : K \longrightarrow K'$, если существуют такие конфигурации K_1 и K_2 МТ M, что K_1 продолжает K, K_2 продолжает K' и $M : K_1 \longmapsto K_2$.

Менее формально, $M : K \longrightarrow K'$ означает, что МТ M перерабатывает K в K' и не добавляет символов слева, а $M : K \longmapsto K'$ означает, что M не добавляет символов ни слева, ни справа и перерабатывает K в K'.

Пример 4.7.2.

Возвращаясь к примеру 4.7.1, видим, что

Пра$_*$: $q \wedge | * 1 | | * \longmapsto \wedge | ! * 1 | | *$.

Вообще, если слово A из E^* не содержит e, то

Пра$_e$: $qaAe \longmapsto aA!e$

для любого a из E, вне зависимости от того, совпадает a с e или нет.

Теорема 4.18. *Существуют следующие машины Тьюринга:*

Пра$_e$ (e из E), $\underline{\text{Лев}_e}$ (e из E), $\overline{\text{Лев}_e}$ (e из E), Зам^a_b (a, b из E), Коп, Пер, Нул, Срв, Огр, $\overline{\text{Нул}}$, Уве,

начальное состояние которых есть q и которые удовлетворяют условиям:

1) Пра$_e$: $qaAe \longmapsto aA!e$ (a в E, A в $(E \setminus \{e\})^*$);

2) $\underline{\text{Лев}_e}$: $eAqa \longmapsto !eAa$ (a в E, A в $(E \setminus \{e\})^*$);

3) $\overline{\text{Лев}_e}$: $eAqa \longmapsto !e \wedge \cdots \wedge$ (a в E, A в $(E \setminus \{e\})^*$);

4) $\underline{\text{Нул}}$: $qd|^n * \longmapsto !d * \wedge^n$;

5) $\overline{\text{Нул}}$: $qd|^n * \longmapsto !d\wedge^{n+1}$;

6) Уве: $qd|^n * \wedge \longmapsto !d|^{n+1} *$;

7) Зам^a_b : $qa \longmapsto !b$;

8) Огр: $q \wedge A \wedge \longmapsto ! \wedge A1$ (A в $\{|, *\}^*$);

9) Коп: $q \wedge A1 \longrightarrow ! \wedge A1A$ (A в $\{|, *\}^*$);

10) Срв: $q \wedge |^n * |^m * \longmapsto !a\wedge^{n+m+2}$, где $a = 1$, если $n < m$, $a = 2$, если $m = n$, $a = 3$, если $n > m$;

11) Пер : $q \wedge |^n * |^m * \longmapsto ! \wedge |^m * |^n *$.

Доказательство.

1). Содержится в примерах 4.7.1, 4.7.2.

2). Лев_e имеет две строки, помеченные начальным состоянием q и состоянием q_1.

В первой строке во всех столбцах стоит Lq_1, во второй строке во всех столбцах, кроме помеченного e, стоит L, а в столбце, помеченном e, стоит !.

3). $\overline{\text{Лев}}_e$ имеет две строки, помеченные начальным q и состоянием q_1.

В первой строке во всех столбцах стоит $\wedge Lq_1$, во второй строке во всех столбцах, кроме помеченного e, стоит $\wedge L$, а в столбце, помеченном e, стоит !.

4). В качестве Нул выбираем следующую МТ:

Нул	$\wedge$	$\mid$	$*$	1	2	3
q	Rq_1		Rq_1			
q_1		R	$\wedge Lq_2$			
q_2	Rq_3	$\wedge L$	Rq_3			
q_3	$*L!$					

Рассмотрим конфигурацию $qd|^n *$. Так как Нул$(d, q) = dRq_1$, то эта конфигурация перерабатывается Нул в конфигурацию $dq_1|^n *$. Так как Нул$(|\,, q_1) = |\,Rq_1$, то эта конфигурация перерабатывается в конфигурацию $d|^n q_1 *$. Так как Нул$(*, q_1) = \wedge Lq_2$, то эта конфигурация перерабатывается в $d|^{n-1}q_2|\wedge$ при $n > 0$ и в $q_2 d\wedge$ при $n = 0$. Так как Нул$(|\,, q_2) = \wedge Lq_2$, то эта конфигурация равна или перерабатывается в $q_2 d \wedge^{n+1}$. Так как Нул$(d, q_2) = dRq_3$, то эта конфигурация перерабатывается в $dq_3 \wedge^{n+1}$. Так как Нул$(\wedge, q_3) = *L!$, то эта конфигурация перерабатывается в $!d * \wedge^n$.

5). В качестве $\overline{\text{Нул}}$ выбираем следующую МТ:

$\overline{\text{Нул}}$	$\wedge$	$\mid$	$*$	1	2	3
q	Rq_1		Rq_1			
q_1		R	$\wedge Lq_2$			
q_2	!	$\wedge L$	!			

6). В качестве Уве выбираем следующую МТ:

Уве	$\wedge$	$\mid$	$*$	1	2	3
q	Rq_1		Rq_1			
q_1	$*Lq_2$	R	$\mid R$			
q_2	!	L	!			

Рассмотрим конфигурацию $qd|^n * \wedge$. Так как Уве$(d,q) = dRq_1$ и Уве$(|\,,q_1) = |\,Rq_1$, то эта конфигурация перерабатывается в $d|^n q_1 * \wedge$. Так как Уве$(*,q_1) = |\,Rq_1$, то эта конфигурация перерабатывается в $d|^{n+1} q_1 \wedge$. Так как Уве$(\wedge,q_1) = *Lq_2$ и Уве$(|\,,q_2) = |\,Lq_2$, то эта конфигурация далее перерабатывается в $q_2 d|^{n+1} *$. Так как Уве$(d,q_2) = dC!$, то окончательно эта конфигурация перерабатывается в $!d|^{n+1} *$.

7). В качестве Зам^a_b выбираем такую МТ с одной строкой, что

$$\text{Зам}^a_b(a,q) = bC!.$$

8). В качестве Огр выбираем следующую МТ:

Огр	$\wedge$	$\vert$	$*$	1	2	3
q	Rq_1					
q_1	$1Lq_2$	R	R			
q_2	!	L	L			

Так как Огр$(\wedge,q) = \wedge Rq_1$ и Огр$(a,q) = aRq_1$ для a из $\{|\,,*\}$, то $q\wedge A\wedge$ перерабатывается Огр в $\wedge Aq_1 \wedge$. Так как

Огр$(\wedge,q_1) = 1Lq_2$ и Огр$(a,q_2) = aLq_2$,

то эта конфигурация перерабатывается в $q_2 \wedge A1$. Так как Огр$(\wedge,q_2) = \wedge C!$, то эта конфигурация перерабатывается в $! \wedge A1$.

9). В качестве Коп выбираем следующую МТ:

Коп	$\wedge$	$\vert$	$*$	1	2	3
q	R	$2Rq_1$	$3Rq_*$	Ls		
q_1	$\vert\,Lr$	R	R	R		
q_*	$*Lr$	R	R	R		
r		L	L	L	$\vert Rq$	$*Rq$
s	!	L	L			

Рассмотрим сначала случай, когда A пусто. В этом случае Коп перерабатывает $q \wedge 1$ последовательно в $\wedge q1, s \wedge 1, ! \wedge 1$, что и требуется.

Пусть теперь, только для определенности A есть $|**|$. Тогда Коп перерабатывает $q \wedge |**| 1 \wedge \wedge \wedge \wedge$ последовательно в

$$\wedge 2**|\,1q_1 \wedge\wedge\wedge\wedge,\quad \wedge r2**|\,1|\wedge\wedge\wedge,\quad \wedge|\,q**|\,1|\wedge\wedge\wedge,$$

$$\wedge|\,3*|\,1|\,q_* \wedge\wedge\wedge,\quad \wedge|\,r3*|\,1|*\wedge\wedge,\quad \wedge|*q*|\,1|*\wedge\wedge,$$

$$\wedge|*3|\,1|*q_*\wedge\wedge,\quad \wedge|*r3|\,1|**\wedge,\quad \wedge|**q|\,1|**\wedge,$$

$$\wedge|**21|**q_1\wedge,\quad \wedge|**r21|**|,\quad \wedge|**|q1|**|,$$

$$s \wedge |\ {*}{*}|\,1|\ {*}{*}|, \quad !\wedge |\ {*}{*}|\,1|\ {*}{*}|.$$

В общем случае надо добавить столько пустых символов, какова длина слова A. Надо заметить, что Коп перерабатывает $\wedge A_1 qaA_2 1A_1\wedge$, где A_1, A_2 из $\{|\,,*\}^*$ и a из $\{|\,,*\}$, в $\wedge A_1 aqA_2 1A_1 a$ и перерабатывает $q\wedge A1\wedge^n$, где n равно длине слова A, в $!\wedge A1A$.

10). В качестве Срв выбираем следующую МТ:

Срв	$\wedge$	$\|$	$*$	1	2	3
q	R	$1Rr$	Rq_0			
r		R	Rs			
s		R	Lt	$\|Lt$		
t		$1Lt_1$	Rt_2			
t_1		L	L	$\|Rq$		
t_2		R	$\wedge Lq_3$			
q_0		Rq_5	$\wedge Lq_2$	$\|Rq_4$		
q_1	1!	$\wedge L$	$\wedge L$			
q_2	2!	$\wedge L$	$\wedge L$			
q_3	3!	$\wedge L$	$\wedge L$	$\wedge L$		
q_4		R	$\wedge Lq_2$			
q_5		R	$\wedge Lq_1$	$\|R$		

Рассмотрим конфигурацию $q\wedge *|^m*$ при $m > 0$. Она перерабатывается в конфигурацию $\wedge * q_0|^m*$ и затем в конфигурации $\wedge * |^m q_5 *$ и $!1\wedge^{m+2}$. Рассмотрим теперь конфигурацию $q\wedge|^n{*}{*}$ при $n > 0$. Она перерабатывается Срв в конфигурацию $\wedge 1|^{n-1} * s *$. Далее последовательно получаем:

$$\wedge 1|^{n-1} t{*}{*}, \quad \wedge 1|^{n-1} * t_2 *, \quad \wedge 1|^{n-1} q_3 * \wedge, \quad !3\wedge^{n+2}.$$

Рассмотрим теперь конфигурацию $q\wedge{*}{*}$. Срв перерабатывает ее в

$$\wedge q{*}{*}, \quad \wedge * q_0 *, \quad \wedge q_2 * \wedge, \quad !2\wedge\wedge.$$

Таким образом, в рассмотренных случаях Срв работает нужным образом.

Пусть теперь $n, m > 0$. Конфигурация $q\wedge|^n*|^m*$ перерабатывается Срв в $\wedge 1r|^{n-1}*|^m*$ и в $\wedge 1|^{n-1}*|^{m-1}t|*$. Затем получаем $\wedge|\,q|^{n-1}*|^{m-1}1*$. Используя индукцию, получаем конфигурации

$$\wedge|^l q|^{n-l} * |^{m-l} 1|^{l-1} *$$

для $l = 1, 2, \ldots$ Рассмотрим теперь отдельно три случая.

Случай 1. $n < m$. Получаем конфигурацию $\wedge|^n q * |^{m-n} 1|^{n-1} *$. Затем получаем $\wedge|^n * q_0|^{m-n}1|^{n-1}*$ и $\wedge|^n * |^m q_5 *$. Наконец, получаем $!1\wedge^{n+m+2}$.

Случай 2. $n > m$. Получаем конфигурацию $\wedge|^m q|^{n-m} * 1|^{m-1}*$. Затем получаем конфигурации $\wedge|^m 1|^{n-m-1} * s1|^{m-1}*$ и $\wedge|^m 1|^{n-m-1} t * |^m *$. Далее получаем $\wedge|^m 1|^{n-m-1} * |^m t_2 *$ и $!3\wedge^{n+m+2}$.

Случай 3. $n = m$. Рассматривается аналогично.

11). Пер задается аналогичной таблицей:

Пер	$\wedge$	$\vert$	$*$	1	2	3
q	R	$1Rr$	Rq_0			
r		R	Rs			
s		R	Lt	$\vert Lt$		
t		$1Lt_1$	$\vert Lt_2$			
t_1		L	L	$\vert Rq$		
t_2		L		$*Lq_2$		
q_0		R	Lq_1	$\vert Lq_1$		
q_1		$*Lq_2$	Lq_2			
q_2	$!$	L	$\vert L$			

Заметим, что за исключением клетки, в которой записано Пер $(*, t)$, первые пять строк у Срв и Пер совпадают. Поэтому Пер перерабатывает конфигурацию $q\wedge|^n * |^m *$ тоже в $\wedge|^l q|^{n-l} * |^{m-l}1|^{l-1}$ для $l = 1, 2, \ldots$ Рассмотрим отдельно несколько случаев.

Случай 1. $0 < n \le m$. Получаем конфигурацию $\wedge|^n q * |^{m-n} 1|^{n-1} *$.

Далее получаем $\wedge|^n * |^{m-n-1} q_1 |^{n+1} *$ или $\wedge|^n q_1 * |^n *$ при $n = m$. Окончательно получаем $!\wedge|^m * |^n *$.

Случай 2. $n > m > 0$. Получаем конфигурации $\wedge|^m q|^{n-m} * 1|^{m-1}*$, затем $\wedge|^m 1|^{n-m-1} * s1|^{m-1}*$ и $\wedge|^m 1|^{n-m-1} t * |^m *$. Окончательно получаем $\wedge|^m t_2 1|^n *$ и $!\wedge|^m * |^n *$.

Случай 3. $m > n = 0$. Последовательно получаем конфигурации:

$$q\wedge * |^m *, \quad \wedge * q_0|^m *, \quad \wedge * |^m q_0 *, \quad \wedge * |^{m-1} q_1 | *, \quad !\wedge|^m * *.$$

Случай 4. $n > m = 0$. Последовательно получаем конфигурации:

$$q\wedge|^n * *, \quad \wedge 1|^{n-1} * s *, \quad \wedge 1|^{n-1} t * *, \quad \wedge t_2 1|^n *, \quad !\wedge * |^n *.$$

Случай 5. $n = m = 0$. Последовательно получаем конфигурации:

$$q\wedge * *, \quad \wedge * q_0 *, \quad \wedge q_1 * *, \quad q_2 \wedge * *, \quad !\wedge * *.$$

∎

Теорема 4.19. *Если A из E^*, M — МТ, K_1 ,K_2 — конфигурации M, $M : K_1 \longmapsto K_2$, то $M : AK_1 \longmapsto AK_2$ и $M : K_1A \longmapsto K_2A$.*

Доказательство. Следует из определений 4.7.5 и 4.7.6. ∎

Определение 4.7.7. *Пусть n — конец ленты δ. Конфигурация $\langle q, i, \delta\rangle$ определяет слово Поста $\Pi(q, i, \delta)$ следующим образом.*

Если $i > 0$ и $\delta(0)$ — пустой символ, пусть $\Pi(q, i, \delta)$ есть $\Pi(q, i - 1, \delta_1)$, где $\delta_1(j) = \delta(j + 1)$ для $j = 0, \ldots, n - 1$ и $n - 1$ — конец ленты δ_1. Если $\delta(n)$ — пустой символ и $i < n$, то $\Pi(q, i, \delta) = \Pi(q, i, \delta_1)$, где $\delta_1(j) = \delta(j)$ для $j = 0, 1, \ldots, n-1$ и $n-1$ — конец ленты δ_1. Если $n = 0$, то $\Pi(q, 0, \delta)$ есть $hq\delta(0)h$. Если $n > 0$ и $\delta(n)$ — непустой символ, то $\Pi(q, 0, \delta)$ есть $hq\delta(0)\ldots\delta(n)h$. Если $\delta(0)$ — непустой символ и $n > 0$, то $\Pi(q, n, \delta)$ есть $h\delta(0)\ldots\delta(n-1)q\delta(n)h$. Если $0 < i < n$ и $\delta(0)$, $\delta(n)$ — непустые символы, то $\Pi(q, i, \delta)$ есть

$$h\delta(0)\ldots\delta(i-1)q\delta(i)\ldots\delta(n)h.$$

Менее формально, в слове Поста сохраняется только наименьший связный кусок ленты, содержащий обозреваемую ячейку и все непустые ячейки.

Теорема 4.20. *Пусть K — конфигурация МТ M.*

$$\Pi(M(K)) = M(\Pi(K)),$$

если конфигурация $M(K)$ определена.

Доказательство. Получается простым сравнением определений путем разбора тринадцати возможных случаев. ∎

4.8 Операции над машинами Тьюринга

Через $Q_1 \bigcup Q_2$ обозначается объединение множеств Q_1 и Q_2. Это значит, что элементами $Q_1 \bigcup Q_2$ являются все элементы Q_1 и все элементы Q_2.

Определение 4.8.1. *Пусть M_1 и M_2 — машины Тьюринга с множествами состояний Q_1 и Q_2 и начальными состояниями q_1 и q_2. Если Q_1 и Q_2 не имеют общих элементов, то $M_1 * M_2$ — это такая машина Тьюринга с множеством состояний $Q_1 \bigcup Q_2$ и начальным состоянием q_1, что $(M_1 * M_2)(e, q) = M_1(\mathrm{e}, q)$, если q из Q_1 и третий элемент тройки $M_1(\mathrm{e}, q)$ не является заключительным состоянием; $(M_1 * M_2)(\mathrm{e}, q)$ получается из $M_1(\mathrm{e}, q)$ заменой ! на q_2, если q из Q_1*

и третий элемент тройки есть !, $(M_1 * M_2)(\mathrm{e}, q) = M_2(\mathrm{e}, q)$, *если* q *из* Q_2. *Если* Q_1 *и* Q_2 *имеют общие элементы, то выбираем некоторое такое множество символов* Q_3, *что* Q_3 *и* Q_1 *не имеют общих элементов и существует одно-однозначное отображение* f *множества* Q_2 *на множество* Q_3. *Через* $f(M_2)$ *обозначаем такую МТ с начальным состоянием* $f(q_2)$, *что множество состояний* $f(M_2)$ *есть* Q_3 *и* $(f(M_2))(\mathrm{e}, f(q)) = s\mathrm{P}f(r)$ *для любых таких* s, e *из* E, P *из* $\{R, C, L\}$ *и* q, r *из* Q_2, *что* $M_2(\mathrm{e}, q) = s\mathrm{P}r$. *В этом случае в качестве* $M_1 * M_2$ *выбираем* $M_1 * f(M_2)$. $M_1 * M_2$ *называется композицией МТ* M_1 *и* M_2.

Менее формально, $f(M_2)$ получается из M_2 заменой в каждой строке метки q на метку $f(q)$ и заменой в каждой клетке таблицы q на $f(q)$ для всех q из Q_2. Если Q_1 и Q_2 не имеют общих элементов, то таблица, задающая $M_1 * M_2$, получается добавлением к таблице, задающей M_1, в которой заключительное состояние ! заменено на начальное состояние q_2 МТ M_2, таблицы, задающей M_2.

Пусть П - заключительное СП $hA!sBh$ (A, B из E^*, s из E). Заменим в П заключительное состояние на начальное состояние q МТ M. Полученное слово Поста $hAqsBh$ обозначим через Π_M. Аналогично, через K_M обозначаем конфигурацию $\langle q, i, \delta \rangle$, если K есть конфигурация $\langle !, i, \delta \rangle$.

Теорема 4.21. *Пусть* M_1 *и* M_2 *— МТ.*

а. Если П *— СП для* M_1*;* Π_2, Π_3 *— заключительные СП;* $\Pi_1 \Longrightarrow^*_{M_1} \Pi_2$*;* $(\Pi_2)_{M_2} \Longrightarrow^*_{M_2} \Pi_3$, *то* $\Pi_1 \Longrightarrow^*_{M_1 * M_2} \Pi_3$.

б. Если K_1 *— конфигурация* M_1, $M_1 : K_1 \longmapsto K_2$, $M_2 : (K_2)_{M_2} \longmapsto K_3$, *то* $(M_1 * M_2) : K_1 \longmapsto K_3$.

Доказательство. Очевидным образом следует из определений. ∎

Таким образом, композиция МТ перерабатывает Π_1 в Π_3, если первая МТ перерабатывает Π_1 в заключительное СП, а вторая МТ — это СП, сделанное начальным для нее, в Π_3. Это свойство имеет место и для конфигураций.

Пример 4.8.1.

Построим МТ $\text{Выб}^3_{1,1}$, удовлетворяющую условию:
$\text{Выб}^3_{1,1}$: $q \wedge |^n * |^m * |^l * \longrightarrow ! \wedge |^n * |^n *$.
В качестве $\text{Выб}^3_{1,1}$ возьмем:
$\text{Пра}_* * \text{Пра}_* * \overline{\text{Нул}} * \overline{\text{Лев}}_* * \text{Зам}^*_\wedge * \text{Лев}_\wedge * \text{Огр} *$
$\text{Коп} * \text{Пра}_1 * \text{Зам}^1_* * \text{Пра}_\wedge * \text{Зам}^\wedge_* * \text{Лев}_\wedge$.

В самом деле,
Пра$_*$: $q\wedge|^n*|^m*|^l*\longrightarrow\wedge|^n!*|^m*|^l*$,
Пра$_*$: $\wedge|^n q*|^m*|^l*\longrightarrow\wedge|^n*|^m!*|^l*$,
$\overline{\text{Нул}}$: $\wedge|^n*|^m q*|^l*\longrightarrow\wedge|^n*|^m!*$,
$\overline{\text{Лев}_*}$: $\wedge|^n*|^m q*\longrightarrow\wedge|^n!*$,
Зам$^*_\wedge$: $\wedge|^n q*\longrightarrow\wedge|^n!\wedge$,
Лев$_\wedge$: $\wedge|^n q\wedge\longrightarrow!\wedge|^n\wedge$,
Огр: $q\wedge|^n\wedge\longrightarrow!\wedge|^n 1$,
Коп: $q\wedge|^n 1\longrightarrow!\wedge|^n 1|^n$,
Пра$_1$: $q\wedge|^n 1|^n\longrightarrow\wedge|^n!1|^n$,
Зам$^1_*$: $\wedge|^n q1|^n\longrightarrow\wedge|^n!*|^n$,
Пра$_\wedge$: $\wedge|^n q*|^n\longrightarrow\wedge|^n*|^n!\wedge$,
Зам$^\wedge_*$: $\wedge|^n*|^n q\wedge\longrightarrow\wedge|^n*|^n!*$,
Лев$_\wedge$: $\wedge|^n*|^n q*\longrightarrow!\wedge|^n*|^n*$.
Аналогично, но проще строится Выб$^2_{1,1}$, удовлетворяющая условию:
Выб$^2_{1,1}$: $q\wedge|^n*|^m*\longrightarrow!\wedge|^n*|^n*$.
Надо в Выб$^3_{1,1}$ начальный кусок
Пра$_*$∗Пра$_*$ ∗ $\overline{\text{Нул}}$ ∗ $\overline{\text{Лев}_*}$
заменить на
Пра$_*$ ∗ $\overline{\text{Нул}}$.
Построим теперь МТ Выб$^3_{1,2}$, удовлетворяющую условию:
Выб$^3_{1,2}$: $q\wedge|^n*|^m*|^l*\longrightarrow!\wedge|^n*|^m*$.
В качестве Выб$^3_{1,2}$ возьмем Пра$_*$∗Пра$_*$ ∗ $\overline{\text{Нул}}$∗Лев$_\wedge$
В самом деле,
Пра$_*$∗Пра$_*$: $q\wedge|^n*|^m*|^l*\longrightarrow\wedge|^n*|^m!*|^l*$,
$\overline{\text{Нул}}$: $\wedge|^n*|^m q*|^l*\longrightarrow\wedge|^n*|^m!*$,
Лев$_\wedge$: $\wedge|^n*|^m q*\longrightarrow!\wedge|^n*|^m*$.

Пример 4.8.2.

Построим МТ Пер$_*$, удовлетворяющую условию:
Пер$_*$: $q*|^n*|^m*\longrightarrow!*|^m*|^n*$.
Легко видеть, что в качестве Пер$_*$ можно взять
Зам$^*_\wedge$∗Пер∗Зам$^\wedge_*$.

Теорема 4.22. *Пусть i, n — натуральные числа, $n>0$, $0<i\le n$. Существует МТ Выб^n_i, удовлетворяющая условию:*
Выб^n_i: $q\wedge|^{x_1}*\cdots*|^{x_n}*\longrightarrow!\wedge|^{x_i}*$.

Доказательство. Если $i=1$, то Выб^n_i есть

$$(\text{Пра}_*)^n*(\overline{\text{Лев}_*})^{n-1}*\text{Лев}_\wedge.$$

Далее используем индукцию по i. Если $i > 2$, то применив

$$(\text{Пра}_*)^{i-2} * \text{Пер}_* * \text{Лев}_\wedge,$$

сведем дело к случаю с меньшим i. Если $i = 2$, то применив Пер, сведем дело к случаю с меньшим i. ∎

Теорема 4.23. *Пусть n — положительное натуральное число, а i и j — положительные натуральные числа, не превосходящие n.*

Существует МТ Выб$^n_{i,j}$, удовлетворяющая условию:

Выб$^n_{i,j}$: $q \wedge |^{x_1} * \cdots * |^{x_n} * \longrightarrow ! \wedge |^{x_i} * |^{x_j} *$.

Доказательство. Если $i = j = 1$ и $n > 3$, то применив

$$(\text{Пра}_*)^n * (\overline{\text{Лев}_*})^{n-3} * \text{Лев}_\wedge,$$

сведем дело к случаю $n = 3$. Выб$^3_{1,1}$ и Выб$^3_{1,2}$ построены в примере 4.8.1.

Индукцией по сумме $i + j$ строим Выб$^n_{i,j}$. Случай $i = j = 1$ уже разобран. Если $i = 1$ и $j = 2$, то в качестве Выб$^n_{1,2}$ можно взять

$$(\text{Пра}_*)^n * (\overline{\text{Лев}_*})^{n-2} * \text{Лев}_\wedge,$$

Если $j = 1$ и $i = 2$, то в качестве Выб$^n_{2,1}$ можно взять

$$(\text{Пра}_*)^n * (\overline{\text{Лев}_*})^{n-2} * \text{Лев}_\wedge * \text{Пер}.$$

Если ни i, ни j не равно 1 и $i \le j$, то, применив $(\text{Пра}_*)^{i-2} * \text{Пер}_* * \text{Лев}_\wedge$ или Пер при $i = 2$, сведем дело к случаю с меньшим i и тем же или меньшим j. Если же $1 < j \le i$, аналогичным образом сведем дело к случаю с меньшим j и тем же или меньшим i. Если $i = 1$ и $j > 2$, то, применив $(\text{Пра}_*)^{j-2} * * \text{Пер}_* * \text{Лев}_\wedge$, сведем дело к случаю с тем же i, но меньшим j. Если $j = 1$ и $i > 2$, аналогичным образом сведем дело к случаю с тем же j, но меньшим i. ∎

Определение 4.8.2. *Пусть M_1, M_2, M_3 — МТ с начальными состояниями q_1, q_2, q_3, множества состояний которых Q_1, Q_2, Q_3 попарно не имеют общих элементов. Выберем q так, что q не содержится ни в Q_1, ни в Q_2, ни в Q_3.*

Пусть M_0 — МТ, задаваемая таблицей:

M	$\wedge$	$\vert$	$*$	*1*	*2*	*3*
q				q_1	q_2	q_3

Ветв(M_1, M_2, M_3) *называется МТ* M *с начальным состоянием* q *и множеством состояний* $Q = \{q\} \bigcup Q_1 \bigcup Q_2 \bigcup Q_3$ *такая, что для любых* s *из* E *и* r *из* Q:

$M(s,r) = M_0(s,r)$, *если* $r = q$,

$M(s,r) = M_1(s,r)$, *если* r *из* Q_1,

$M(s,r) = M_2(s,r)$, *если* r *из* Q_2,

$M(s,r) = M_3(s,r)$, *если* r *из* Q_3.

Если какие-то два из множеств Q_1, Q_2, Q_3 *имеют общие элементы, то выбираем такое* Q' *и одно-однозначное отображение* f_2 Q_2 *на* Q', *что* Q' *и* Q_1 *не имеют общих элементов, а затем выбираем* Q'' *и одно-однозначное отображение* f_3 Q_3 *на* Q'' *такое, что* Q'' *и* $(Q_1 \bigcup Q')$ *не имеют общих элементов. В качестве Ветв*(M_1, M_2, M_3) *берем Ветв*$(M_1, f_2(M_2), f_3(M_3))$.

Теорема 4.24. *Пусть*

$$M_1 : \langle q_1, i, \delta \rangle \longmapsto \langle !, i_1, \delta_1 \rangle,$$

$$M_2 : \langle q_2, i, \delta \rangle \longmapsto \langle !, i_2, \delta_2 \rangle,$$

$$M_3 : \langle q_3, i, \delta \rangle \longmapsto \langle !, i_3, \delta_3 \rangle.$$

Тогда если $j \in \{1, 2, 3\}$ *и* $\delta(i) = j$, *то* $M : \langle q, i, \delta \rangle \longmapsto \langle !, i_j, \delta_j \rangle$.

Доказательство. Очевидным образом следует из определения. ∎

Определение 4.8.3. *Пусть* η, χ *из* $\{0, 1\}$. *Пусть* M_1, M_2, M_3, M_4 — *МТ. Под* $M_4 *$*Ветв*$((M_1)^\eta, (M_2)^\chi, M_3)$ *понимается следующая МТ.*

Если q *не принадлежит* $Q_4 \bigcup Q_1 \bigcup Q_2 \bigcup Q_3$; *множества состояний* Q_4, Q_1, Q_2, Q_3 *МТ* M_4, M_1, M_2, M_3 *попарно не имеют общих элементов;* q_4, q_1, q_2, q_3 *являются начальными состояниями* M_4, M_1, M_2, M_3; M_0 *обозначает МТ, описанную в определении 4.8.2, то*

$$M_4 * \text{Ветв}((M_1)^\eta, (M_2)^\chi, M_3)$$

есть такая МТ M *с начальным состоянием* q_4 *и множеством состояний* $Q = \{q\} \bigcup Q_1 \bigcup Q_2 \bigcup Q_3 \bigcup Q_4$, *что для любых* s *из* E *и* r *из* Q *имеем:*

$M(s,r) = M_4(s,r)$, *если в тройке* $M_4(s,r)$ *третий элемент не является заключительным состоянием и* r *из* Q_4;

$M(s,r)$ *получается из* $M_4(s,r)$ *заменой* ! *на* q, *если* r *из* Q_4 *и третий элемент тройки* $M_4(s,r)$ *есть* !;

$M(s,r) = M_0(s,r)$, *если* $r = q$;

$M(s,r) = M_1(s,r)$, если r из Q_1 и либо $\eta = 1$, либо третий элемент тройки $M_1(s,r)$ отличен от !;

$M(s,r)$ получается из $M_1(s,r)$ заменой ! на q_4, если r из Q_1, $\eta = 0$ и третий элемент тройки $M_1(s,r)$ есть !;

$M(s,r) = M_2(s,r)$, если r из Q_2 и либо $\chi = 1$, либо третий элемент тройки $M_2(s,r)$ отличен от !;

$M(s,r)$ получается из $M_2(s,r)$ заменой ! на q_4, если r из Q_2, $\chi = 0$ и третий элемент тройки $M_2(s,r)$ есть !;

$M(s,r) = M_3(s,r)$, если r из Q_3.

Если какие-то два из множеств Q_1, Q_2, Q_3, Q_4 имеют общие элементы, то для $j = 1,2,3$ выбираем такое множество Q'_j, которое не имеет общих элементов с Q_4 при $j = 1$, с $Q_4 \bigcup Q'_1$ при $j = 2$ и с $Q_4 \bigcup Q'_1 \bigcup Q'_2$ при $j = 3$ и для которого существует одно-однозначное отображение f_j множества Q_j на Q'_j. В качестве

$$M_4 * \text{Ветв}((M_1)^\eta, (M_2)^\chi, M_3)$$

выбираем

$$M_4 * \text{Ветв}((f_1(M_1))^\eta, (f_2(M_2))^\chi, f_3(M_3)).$$

Теорема 4.25. *Пусть $M = M_4 * Ветв((M_1)^\eta, (M_2)^\chi, M_3)$ и*

$$M_4 : \langle q_4, i, b\rangle \longmapsto \langle !, i', b'\rangle.$$

а. Пусть $\eta = 0$, $\chi = 1$.

Если $b'(i') = 1$, $M_1 : \langle q_1, i', b'\rangle \longmapsto \langle !, i_1, b_1\rangle$, то $M : \langle q_4, i, b\rangle \longmapsto \langle q_4, i_1, b_1\rangle$.

Если $b'(i') = 2$, $M_2 : \langle q_2, i', b'\rangle \longmapsto \langle !, i_2, b_2\rangle$, то $M : \langle q_4, i, b\rangle \longmapsto \langle !, i_2, b_2\rangle$.

Если $b'(i') = 3$, $M_3 : \langle q_3, i', b'\rangle \longmapsto \langle !, i_3, b_3\rangle$, то $M : \langle q_4, i, b\rangle \longmapsto \langle !, i_3, b_3\rangle$.

б. Пусть $\chi = 0$, $\eta = 1$.

Если $b'(i') = 1$, $M_1 : \langle q_1, i', b'\rangle \longmapsto \langle !, i_1, b_1\rangle$, то $M : \langle q_4, i, b\rangle \longmapsto \langle !, i_1, b_1\rangle$.

Если $b'(i') = 2$, $M_2 : \langle q_2, i', b'\rangle \longmapsto \langle !, i_2, b_2\rangle$, то $M : \langle q_4, i, b\rangle \longmapsto \langle q_4, i_2, b_2\rangle$.

Если $b'(i') = 3$, $M_3 : \langle q_3, i', b'\rangle \longmapsto \langle !, i_3, b_3\rangle$, то $M : \langle q_4, i, b\rangle \longmapsto \langle !, i_3, b_3\rangle$.

Доказательство. Следует из предыдущих теорем и определений. ∎

В дальнейшем вместо Ветв$((M_1)^\eta, (M_2)^\chi, M_3)$ будем писать

$$\text{Ветв}\begin{cases} (M_1)^\eta \\ (M_2)^\chi \\ M_3. \end{cases}$$

Если $\eta = 1$, будем опускать η, если $\chi = 1$, будем опускать χ.

Менее формально, $M = M_4 *$Ветв$((M_1)^\eta, (M_2)^\chi, M_3)$ работает сначала как M_4. Если M_4 останавливается, обозревая 1, то M далее работает как M_1. Если M_1 останавливается, то M тоже останавливается, если $\eta = 1$, или возвращается в начальное состояние M_4, если $\eta = 0$. Если M_4 останавливается, обозревая 2, то M далее работает как M_2. Если M_2 останавливается, то M тоже останавливается, если $\chi = 1$, или возвращается в начальное состояние M_4, если $\chi = 0$. Если M_4 останавливается, обозревая 3, то M далее работает как M_3. Если M_3 останавливается, то M тоже останавливается.

4.9 Тьюрингова вычислимость

Для структурированной программы Π будем использовать Y_{Π,x_i} для обозначения $(\Pi(\sigma))(x_i)$ как функций от $\sigma(x_1), \ldots, \sigma(x_n)$ (см. начало §4.5).

Определение 4.9.1. *Скажем, что МТ M с начальным состоянием q моделирует работу структурированной программы Π на наборе переменных $x_1, \ldots, x_n$, если*

а) программа Π не использует переменных, отличных от $x_1, \ldots, x_n$;

б) для любого набора $\langle a_1, \ldots, a_n\rangle$ натуральных чисел

$$M : q \wedge |^{a_1} * \cdots * |^{a_n} * \longrightarrow ! \wedge |^{b_1} * \cdots * |^{b_n} *$$

тогда и только тогда, когда

$$Y_{\Pi,x_1}(a_1, \ldots, a_n) = b_1, \ldots, Y_{\Pi,x_n}(a_1, \ldots, a_n) = b_n;$$

в) для любого состояния σ состояние $\Pi(\sigma)$ определено тогда и только тогда, когда МТ M перерабатывает СП $hq \wedge |^{\sigma(x_1)} * \cdots * |^{\sigma(x_n)} * h$ в заключительное СП.

Теорема 4.26. *Для каждой структурированной программы Π и каждого набора переменных $x_1, \ldots, x_n$, содержащего все переменные, используемые программой Π, существует МТ $M(\Pi)$, моделирующая работу Π на $x_1, \ldots, x_n$.*

Доказательство. Каждая структурированная программа П является последней программой в некоторой последовательности программ, каждая из которых либо является присваиванием, либо получается из предыдущих программ этой же последовательности путем композиции, условного оператора или цикла. Теорему доказываем индукцией по длине такой последовательности. Таким образом, надо рассмотреть случай, когда П — присваивание, а также, допуская, что теорема верна для Π_1 и Π_2, доказать ее для композиции, условного оператора и цикла.

Лемма 4.27. *Пусть* Π_n^j *— это присваивание* $x_n \leftarrow x_n'$ *при* $j = 1$, $x_n \leftarrow 0$ *при* $j = 2$. *В качестве* $M(\Pi_n^j)$ *можно взять следующую МТ:*
Уве при $n = 1, j = 1$; *Нул при* $n = 1, j = 2$;
$(\text{Пра}_*)^{n-1} * \text{Уве} * \text{Лев}_\wedge$ *при* $n > 1$, $j = 1$;
$(\text{Пра}_*)^{n-1} * \text{Нул} * \text{Лев}_\wedge$ *при* $n > 1$, $j = 2$.

Доказательство. Очевидно. ∎

Лемма 4.28. *Пусть* Π_i^j *— это присваивание* $x_i \leftarrow x_i'$ *при* $j = 1$, $x_i \leftarrow 0$ *при* $j = 2$. *В качестве* $M(\Pi_i^j)$ *при* $i < n$ *можно взять следующую МТ:*
$(\text{Пра}_*)^{i-1} * \text{Пер}_* * \text{Лев}_\wedge * M(\Pi_{i+1}^j) * (\text{Пра}_*)^{i-1} * \text{Пер}_* * \text{Лев}_\wedge$ *при* $i > 1$,
$\text{Пер} * M(\Pi_{i+1}^j) * \text{Пер}$ *при* $i = 1$.

Доказательство. В самом деле, при $i > 1$ $(\text{Пра}_*)^{i-1} * \text{Пер}_* * \text{Лев}_\wedge$ переставляет i-ый и $(i+1)$-ый блоки палочек. Затем число палочек в старом i-ом блоке увеличивается на 1 или обнуляется, а затем этот блок возвращается на прежнее место. При $i = 1$ нужная перестановка осуществляется просто программой Пер. ∎

Лемма 4.29. *Пусть* $\Pi(i,j)$ *это присваивание* $x_i \leftarrow x_j$.
а. В качестве $M(\Pi(i,i))$ *можно взять* $\text{Зам}_\wedge^\wedge$.
б. В качестве $M(\Pi(n,n-1))$ *можно взять следующую МТ:*
$(\text{Пра}_*)^{n-1} * \text{Нул} * \text{Пра}_* * \text{Зам}_\wedge^* * \text{Лев}_* * \text{Зам}_1^* * \text{Лев}_* * \text{Зам}_\wedge^* *$
$\text{Коп} * \text{Зам}_*^\wedge * \text{Пра}_1 * \text{Зам}_*^1 * \text{Пра}_\wedge * \text{Зам}_*^\wedge * \text{Лев}_\wedge$.
в. При $i < n-1$ *в качестве* $M(\Pi(n,i))$ *можно взять следующую МТ:*
$(\text{Пра}_*)^{i-1} * \text{Пер}_* * \text{Лев}_\wedge * M(\Pi(n,i+1)) * (\text{Пра}_*)^{i-1} * \text{Пер}_* * \text{Лев}_\wedge$
при $i > 1$ *или*
$\text{Пер} * M(\Pi(n,i+1)) * \text{Пер}$
при $i = 1$.
г. При $i < j < n$ *в качестве* $M(\Pi(j,i))$ *можно взять следующую МТ:*
$(\text{Пра}_*)^{j-1} * \text{Пер}_* * \text{Лев}_\wedge * M(\Pi(j+1,i)) * (\text{Пра}_*)^{j-1} * \text{Пер}_* * \text{Лев}_\wedge$.

д. В качестве $M(\Pi(i-1,i))$ можно взять следующую МТ:

(Пра$_*$)$^{i-2}$∗Пер$_*$∗Лев$_\wedge$ ∗ $M(\Pi(i,i-1))$

при $i>2$ или

Пер∗$M(\Pi(2,1))$

при $i=2$.

е. При $i+1<j$ в качестве $M(\Pi(i,j))$ можно взять следующую МТ:

(Пра$_*$)$^{i-1}$∗Пер$_*$∗Лев$_\wedge$ ∗ $M(\Pi(i+1,j))$ ∗ (Пра$_*$)$^{i-1}$∗Пер$_*$∗Лев$_\wedge$,

если $i>1$ или

Пер∗$M(\Pi(i+1,j))$∗Пер

при $i=1$.

Доказательство.

а. Очевидно.

б. Имеем:

$$(\text{Пра}_*)^{n-1}:\ q\wedge|^{x_1}*\cdots*|^{x_n}*\longmapsto\wedge|^{x_1}*\cdots*|^{x_{n-1}}!*|^{x_n}*,$$

$$\text{Нул}:\ \wedge|^{x_1}*\cdots*|^{x_{n-1}}q*|^{x_n}*\longrightarrow\wedge|^{x_1}*\cdots*|^{x_{n-1}}!**,$$

Выполнив после этого

Пра$_*$∗Зам$^*_\wedge$∗Лев$_*$∗Зам$^*_1$∗Лев$_*$∗Зам$^*_\wedge$∗Коп∗Зам$^\wedge_*$

мы получим:

$$\wedge|^{x_1}*\ldots!*|^{x_{n-1}}1|^{x_{n-1}}.$$

Окончательно

Пра$_1$∗Зам$^1_*$∗Пра$_\wedge$∗Зам$^\wedge_*$∗Лев$_\wedge$

перерабатывает полученную ленту в

$$!\wedge|^{x_1}*\cdots*|^{x_{n-1}}*|^{x_{n-1}}*.$$

в—е. Рассматриваются аналогично. ■

Лемма 4.30. *Пусть МТ M_1 моделирует работу структурированной программы Π_1, а МТ M_2 — структурированной программы Π_2 на наборе переменных $x_1,\ldots,x_n$. Тогда M_1*M_2 моделирует работу $\Pi_1;\Pi_2$ на этом же наборе переменных.*

Доказательство. Следует из определений и теоремы 4.21. ■

Лемма 4.31. *Пусть МТ M_1 моделирует работу структурированной программы Π_1, а МТ M_2 — структурированной программы Π_2 на наборе переменных $x_1, \ldots, x_n$. Пусть M есть следующая МТ:*

$$\textit{Огр} * \textit{Коп} * \textit{Пра}_1 * \textit{Зам}^1_\wedge * \textit{Выб}^n_{i,j} * \textit{Срв} * \textit{Ветв} \begin{cases} \textit{Зам}^1_\wedge * \textit{Лев}_\wedge * N_1 \\ \textit{Зам}^2_\wedge * \textit{Лев}_\wedge * N_2 \\ \textit{Зам}^3_\wedge * \textit{Лев}_\wedge * N_3 \end{cases}$$

Тогда МТ M моделирует на том же наборе переменных работу программы $(*)$:

"если $x_i = \mathrm{x}_j$ то Π_1 иначе Π_2 конец "

в случае, когда N_1 есть M_2, N_2 есть M_1, N_3 есть M_2, и моделирует на том же наборе переменных работу программы $(**)$:

"если $x_i < x_j$ то Π_1 иначе Π_2 конец "

в случае, когда N_1 есть M_1, N_2 есть M_2, N_3 есть M_2.

Доказательство. В самом деле,

$$\text{Огр} * \text{Коп} * \text{Пра}_1 * \text{Зам}^1_\wedge * \text{Выб}^n_{i,j} * \text{Срв}$$

переводит

$$q \wedge |^{a_1} * \cdots * |^{a_n} *$$

в

$$\wedge|^{a_1} * \cdots * |^{a_n} *!e,$$

где $e = 1$, если $a_i < a_j$; $e = 2$, если $a_i = a_j$; $e = 3$, если $a_i > a_j$.

Применив

$$\text{Зам}^e_\wedge * \text{Лев}_\wedge,$$

мы получим ленту

$! \wedge |^{a_1} * \cdots * |^{a_n} *$.

Поэтому, если N_1 есть M_2, N_2 есть M_1, N_3 есть M_2, то при $a_i = a_j$ рассматриваемая МТ M работает как M_1, а при $a_i \neq a_j$ M работает как M_2. Значит, рассматриваемая МТ M моделирует работу программы $(*)$.

Поэтому, если N_1 есть M_1, а N_2 и N_3 есть M_2, то при $a_i < a_j$ рассматриваемая программа работает как M_1, а при $a_i \geq a_j$ M работает как M_2. Значит, M моделирует работу программы $(**)$. ∎

Лемма 4.32. *Пусть МТ M_1 моделирует работу структурированной программы Π_1 на наборе переменных $x_1, \ldots, x_n$. Пусть M есть следующая МТ:*

$$\textit{Огр} * \textit{Коп} * \textit{Пра}_1 * \textit{Зам}^1_\wedge * \textit{Выб}^n_{i,j} * \textit{Срв} * \textit{Ветв} \begin{cases} (\textit{Зам}^1_\wedge * \textit{Лев}_\wedge * N_1)^\eta \\ (\textit{Зам}^2_\wedge * \textit{Лев}_\wedge * N_2)^\chi \\ \textit{Зам}^3_\wedge * \textit{Лев}_\wedge * N_3 \end{cases}$$

Тогда M моделирует на том же наборе работу программы:

"пока $x_i = x_j$ делай Π_1 все "

в случае, когда N_1 и N_3 есть $\textit{Зам}^\wedge_\wedge$, N_2 есть M_1, $\eta = 1$, $\chi = 0$, и моделирует на том же наборе работу программы

"пока $x_i < x_j$ делай Π_1 все "

в случае, когда N_2 и N_3 есть $\textit{Зам}^\wedge_\wedge$, $\eta = 0$, N_1 есть M_1, $\chi = 1$.

Доказательство. Как и в предыдущем случае, мы получим ленту

$q \wedge |^{a_1} * \cdots * |^{a_n} *$,

к которой надо применить машину N_e. Если $\eta = 0$ и $N_1 = M_1$, то мы работаем, пока $a_i < a_j$. Если же $\chi = 0$ и N_2 есть M_1, мы работаем, пока $a_i = a_j$. ∎

Теорема доказана. ∎

Определение 4.9.2. *Скажем,что МТ M с начальным состоянием q вычисляет арифметическую функцию ϕ от n аргументов, если*

а) для любого набора $\langle a_1, \ldots, a_n \rangle$ натуральных чисел

$$M: \ q \wedge |^{a_1} * \cdots * |^{a_n} * \longrightarrow ! \wedge |^{f(a_1, \ldots, a_n)} *$$

тогда и только тогда, когда $\phi(a_1, \ldots, a_n)$ определено;

б) если $\phi(a_1, \ldots, a_n)$ не определено, то M не перерабатывает начальное СП

$$hq \wedge |^{a_1} * \cdots * |^{a_n} * h$$

ни в какое заключительное СП.

Арифметическая функция называется вычислимой по Тьюрингу (Т-вычислимой), если она вычисляется некоторой МТ.

Теорема 4.33. *Для каждой частично рекурсивной функции ϕ существует вычисляющая ее машина Тьюринга.*

Доказательство. По теореме 4.5, каждая частично рекурсивная функция вычисляется некоторой структурированной программой в некоторой переменной при некотором разбиении используемых переменных на основные и рабочие. Пусть n-местная частично рекурсивная функция ϕ вычисляется структурированной программой Π в переменной x_1 с основными переменными

$$x_1, \ldots, x_n$$

и рабочими переменными

$$x_{n+1}, \ldots, x_m.$$

Пусть МТ M моделирует работу Π на наборе переменных

$$x_1, \ldots, x_n, x_{n+1}, \ldots, x_m.$$

Рассмотрим МТ M_1 :
$(\text{Пра}_\wedge * \text{Зам}_*^\wedge)^{m-n} * \text{Лев}_\wedge * M * \text{Выб}_1^m$.
Сравнивая определения, получаем, что M_1 вычисляет ϕ. ∎

4.10 Частичная рекурсивность Т-вычислимых функций

Определение 4.10.1. *Символам $\wedge$, $|$, $*$, 1, 2, 3 приписываем номера 0, 1, 2, 3, 4, 5. Пустому слову приписываем номер 1. Слову $a_0 \ldots a_n$ в алфавите E приписываем номер*

$$\prod_{i=0}^{n} p(i)^{\zeta(a_i)},$$

где $\zeta(a_i)$ обозначает номер символа a_i, а $p(i)$ по-прежнему обозначает i-ое по порядку простое число.

Обращением пустого слова является пустое слово. Обращением слова Aa является слово aA^*, где A^* — обращение слова A.

Например, слово $|| * |*$ имеет номер $2 * 3 * 5^2 * 7 * 11^2$. Обращением этого слова является слово $*| * ||$.

Определение 4.10.2. *Символу ! приписываем номер 0. Если Q — алфавит состояний МТ M, начальному состоянию приписываем номер 1, а остальным состояниям номера от 2 до m, где m — число элементов Q. Слову Поста для M $hArsBh$ приписываем номер $2^{n_1} * 3^{n_2} *$*

$5^{n_3} * 7^{n_4}$, где n_1 — номер обращения слова A, значит, номер слова, получаемого из слова A, если последний его символ сделать первым, предпоследний — вторым и т.д., n_2 — номер состояния r, n_3 — номер символа s из E, n_4 — номер слова B. Номером конфигурации $\langle q, i, \delta\rangle$ с концом n ленты δ называется число $2^{n_1} * 3^{n_2} * 5^{n_3} * 7^{n_4}$, где n_1 — номер слова $\delta(i-1)\ldots\delta(0)$, n_2 — номер состояния q, n_3 — номер символа $\delta(i)$ из E, n_4 — номер слова $\delta(i+1)\ldots\delta(n)$.

Заметим, что номер слова не изменится, если к нему в конце приписать любую последовательность пустых символов, так как пустой символ имеет номер 0. Поэтому, если конфигурация K_1 продолжает конфигурацию K_2, то номер K_1 совпадает с номером K_2. Заметим также, что если с конфигурацией K связывается СП П(К), то K и П(К) имеют одинаковые номера.

Определение 4.10.3. *Пусть M — МТ с множеством состояний Q. Пусть m — число элементов Q.*

Номером тройки $\langle s, O, q\rangle$ назовем число

$$2^{n_1} * 3^{n_2} * 5^{n_3},$$

где n_1 — номер первого элемента этой тройки, n_3 — номер третьего элемента этой тройки, $n_2 = 0$, если второй элемент тройки есть C, $n_2 = 1$, если второй элемент тройки есть L, $n_2 = 2$, если второй элемент тройки есть R.

Если s имеет номер i, $0 \le i \le 5$, а q имеет номер j, $1 \le j \le m$, то номер тройки $M(s,q)$ обозначается через $\zeta_M(i,j)$.

Пусть $\zeta_M(j)$ есть

$$\prod_{i=0}^{5} p(i)^{\zeta_M(i,j)},$$

а ζ_M есть

$$\prod_{j=1}^{m} p(j)^{\zeta_M(j)}.$$

Число $\zeta_M(j)$ называется номером j-той строки МТ M, а число ζ_M — номером МТ M.

Пример 4.10.1.

Рассмотрим МТ Пра$_*$, описанную в примере 4.7.1. Здесь q имеет номер 1, а q_1 — номер 2. Первая строка имеет 6 клеток с номерами

$2^0 * 3^2 * 5^2$, $2^1 * 3^2 * 5^2$, $2^2 * 3^2 * 5^2$, $2^3 * 3^2 * 5^2$, $2^4 * 3^2 * 5^2$, $2^5 * 3^2 * 5^2$, которые равны 225, 450, 900, 1800, 3600, 7200. Поэтому номер первой строки равен

$$2^{225} * 3^{450} * 5^{900} * 7^{1800} * 11^{3600} * 13^{7200}.$$

Вторая строка имеет 6 клеток с номерами $2^0*3^2*5^2$, $2^1*3^2*5^2$, $2^2*3^0*5^0$, $2^3 * 3^2 * 5^2$, $2^4 * 3^2 * 5^2$, $2^5 * 3^2 * 5^2$, которые равны 225, 450, 4, 1800, 3600, 7200. Поэтому номер второй строки — это

$$2^{225} * 3^{450} * 5^{4} * 7^{1800} * 11^{3600} * 13^{7200}.$$

Номер же Пра_* — это $3^{n_1} * 5^{n_2}$, где n_1 — номер первой строки, а n_2 — номер второй строки.

Теорема 4.34. *Пусть M — МТ с начальным состоянием q. Существует такая рекурсивная функция ζ_n от n аргументов, что для любого набора $\langle a_1, \dots, a_n\rangle$ натуральных чисел $\zeta_n(a_1, \dots, a_n)$ есть номер начального слова Поста $hq \wedge |^{a_1} * \cdots * |^{a_n} * h$.*

Доказательство. Индукцией по n. Будем использовать обозначения для простейших функций, введенные в определении 4.2.4.

Базис индукции, $n = 1$. В этом случае рассматриваемое СП есть $hq \wedge |^{a_1} * h$ и его номер есть $2 * 3 * 7^{\psi_1(a_1)}$, где $\psi_1(a)$ есть

$$(\prod_{i=0}^{a-1} p(i)) * (p(a))^2,$$

если $a > 0$, и $\psi_1(0) = 4$. Другими словами,

$$\psi_1(a) = sg(a) * (\prod_{i=0}^{a \frown 1} p(i)) * (p(a)^2) + 4 * \overline{sg}(a).$$

Рекурсивность ψ_1 следует из рекурсивности $\prod_{i=0}^{z} p(i)$, которая легко доказывается, если использовать теорему об ограниченном перемножении (теорема 4.3). Действительно, если $p'(x, y) = p(y)$, то $p' = [p;\ \eta_2^2]$. Далее, $p''(x, z) = \prod_{i=0}^{z} p'(x, i)$ рекурсивна (по теореме 4.3). Наконец,

$$\psi_1(a) = [+;\ [*;\ [*;\ sg,\ [p'';\ \eta_1^1, [(- :);\ \eta_1^1]]], [*;\ p,\ p]],\ [*;\ 4,\ \overline{sg}]](a).$$

Индукционный шаг. Пусть $\zeta_n(a_1, \dots, a_n)$ — рекурсивная функция. Тогда

$$\zeta_{n+1}(a_1, \dots, a_n, a_{n+1}) = 2 * 3 * 7^{\psi_{n+1}(a_1, \dots, a_n, a_{n+1})},$$

где $\psi_{n+1}(a_1,\dots,a_n,a_{n+1})$ — номер слова $|^{a_1} * \dots * |^{a_n} * |^{a_{n+1}} *$. Если $\psi_n(a_1,\dots,a_n)$ — номер слова $|^{a_1} * \dots * |^{a_n} *$, то $\psi_n = [exp;\ \zeta_n, [';[';[';\theta]]]]$ и, значит, рекурсивна. Однако

$$\psi_{n+1}(a_1,\dots,a_n,a_{n+1}) = \psi_n(a_1,\dots,a_n) * (\prod_{i=a_1+\dots+a_n+n+1}^{a_1+\dots+a_n+a_{n+1}+n} p(i)) *$$
$$p(a_1+\dots+a_n+a_{n+1}+n+1) * p(a_1+\dots+a_n+a_{n+1}+n+1).$$

∎

Теорема 4.35. *Существует такая рекурсивная функция $\Theta(m,n)$, что если m — номер МТ, а n — номер незаключительного СП для этой МТ, то $\Theta(m,n)$ — это номер СП, в которое МТ номера m перерабатывает СП номера n за один такт работы. Если же n — номер СП $hA! \wedge Bh$, то $\Theta(m,n) = n$.*

Доказательство. $exp(n,1)$ задает номер состояния, а $exp(n,2)$ задает номер обозреваемого символа. Поэтому $exp(exp(m,exp(n,1)),exp(n,2))$ задает номер тройки, стоящей на пересечении строки номера $exp(n,1)$ и столбца номера $exp(n,2)$ в МТ номера m.

Обозначим через $J_{i+1}(m,n)$ число
$exp(exp(exp(m,exp(n,1)),exp(n,2)),i)$
для $i=0,1,2$. Рассмотрим отдельно три случая, в зависимости от того, равно ли $J_2(m,n)$ числу 0, 1 или 2.

Случай 1. $J_2(m,n)=2$. В этом случае происходит движение вправо. Значит, если n — номер слова Поста $hArsBh$, то новым обозреваемым символом будет первый символ B, если B непусто, или пустой символ, если B пусто. Новое слово B получится из старого отбрасыванием первой буквы. Поэтому номер нового слова B равен

$$\Theta_1(m,n) = \prod_{i=0}^{exp(n,3)} p(i)^{exp(exp(n,3),i+1)}.$$

Новое слово A получается приписыванием к A в конце символа с номером $J_1(m,n)$. Поэтому номер обращения нового слова A есть:

$$\Theta_2(m,n) = 2^{J_1(m,n)} * \prod_{i=1}^{exp(n,0)} p(i)^{exp(exp(n,0),(-:)(i))}.$$

Окончательно, номер нового слова Поста, получающегося из СП номера n за один такт работы МТ номера m, есть

$$\Theta_3(m,n) = 2^{\Theta_2(m,n)} * 3^{J_3(m,n)} * 5^{exp(exp(n,3),0)} * 7^{\Theta_1(m,n)}.$$

Случай 2. $J_2(m,n) = 1$. Рассматривается аналогично. В этом случае происходит движение влево. Обращение нового слова A имеет номер:

$$\Theta_4(m,n) = \prod_{i=0}^{exp(n,0)} p(i)^{exp(exp(n,0),i+1)}.$$

Новое слово B имеет номер:

$$\Theta_5(m,n) = 2^{J_1(m,n)} * \prod_{i=1}^{exp(n,3)} p(i)^{exp(exp(n,3),(-:)(i))}.$$

Окончательно, номер СП, в которое МТ номера m перерабатывает СП номера n, есть

$$\Theta_6(m,n) = 2^{\Theta_4(m,n)} * 3^{J_3(m,n)} * 5^{exp(exp(n,0),0)} * 7^{\Theta_5(m,n)}.$$

Случай 3. $J_2(m,n) = 0$. В этом случае движения не происходит. Поэтому A и B не меняются. Значит, номер СП, в которое МТ номера m перерабатывает СП номера n, есть

$$\Theta_7(m,n) = 2^{exp(n,0)} * 3^{J_3(m,n)} * 5^{J_1(m,n)} * 7^{exp(n,3)}.$$

Итак, получаем, что $\Theta(m,n)$ надо положить $\Theta_3(m,n)$, если $|J_2(m,n) - 2| = 0$, положить $\Theta_6(m,n)$, если $|J_2(m,n) - 1| = 0$, положить $\Theta_7(m,n)$, если $J_2(m,n) = 0$, и положить n в остальных случаях. По теореме о разборе случаев (теорема 4.4), такая $\Theta(m,n)$ будет рекурсивной.

Заметим, что если $exp(n,1) = 0$, то $exp(m, exp(n,1)) = 0$. Поэтому, $J_1(m,n) = J_2(m,n) = J_3(m,n) = 0$. Следовательно, если МТ номера m останавливается, обозревая пустой символ в заключительном СП номера n, то $\Theta(m,n) = n$. ∎

Теорема 4.36. *Существует такая рекурсивная функция $\Theta_8(m,n,i)$, что если m — номер МТ, а n — номер СП для МТ номера m, то $\Theta_8(m,n,i)$ задает номер СП, в которое МТ номера m перерабатывает СП номера n за i тактов работы.*

Доказательство. Надо положить $\Theta_8(m,n,0) = n$ и $\Theta_8(m,n,i+1) = \Theta(m, \Theta_8(m,n,i))$. ∎

Теорема 4.37. *Каждая арифметическая функция, вычисляемая некоторой МТ, является частично рекурсивной.*

Доказательство. Пусть МТ M имеет номер m и вычисляет n-местную арифметическую функцию ϕ. Тогда

$$exp(\Theta_8(m, \zeta_n(a_1, \ldots, a_n), i), 1) = 0$$

тогда и только тогда, когда МТ M переводит начальное СП номера $\zeta_n(a_1, \ldots, a_n)$ в заключительное слово Поста.

Обозначим через $\Theta_9(m, a_1, \ldots, a_n)$ наименьшее из таких i, что

$$exp(\Theta_8(m, \zeta_n(a_1, \ldots, a_n), i), 1) = 0.$$

Ясно,что функция Θ_9 частично рекурсивна. $\Theta_9(m, a_1, \ldots, a_n)$ определено тогда и только тогда, когда $\phi(a_1, \ldots, a_n)$ определено. Понятно, что

$$\Theta_8(m, \zeta_n(a_1, \ldots, a_n), \Theta_9(m, a_1, \ldots, a_n))$$

задает номер заключительного СП. Это СП имеет вид

$$h! \wedge |^{f(a_1, \ldots, a_n)} * h.$$

Обозначим через $\Theta_{10}(m, a_1, \ldots, a_n)$ число

$$exp(\Theta_8(m, \zeta_n(a_1, \ldots, a_n), \Theta_9(m, a_1, \ldots, a_n)), 3).$$

Ясно, что Θ_{10} — частично рекурсивная функция и $\Theta_{10}(m, a_1, \ldots, a_n)$ есть номер слова $|^{\phi(a_1, \ldots, a_n)} *$. Теперь $\phi(a_1, \ldots, a_n)$ есть наименьшее i, что $exp(\Theta_{10}(m, a_1, \ldots, a_n), i) = 2$. Функция

$$\Theta_{11} = M[| - |;\ [exp;\ [\Theta_{10};\ \eta_1^{n+2},\ \eta_2^{n+2}, \ldots, \eta_{n+1}^{n+2}], \eta_{n+2}^{n+2}],\ [';\ [';\ [\theta;\ \eta_1^{n+2}]]]]$$

частично рекурсивна и

$$\phi(a_1, \ldots, a_n) = \Theta_{11}(m, a_1, \ldots, a_n),$$

если $\phi(a_1, \ldots, a_n)$ определено. ∎

Теорема 4.38. *Каждая рекурсивная функция является последней функцией некоторой такой конечной последовательности всюду определенных функций, каждая из которых либо является простейшей, либо получается из предыдущих функций этой же последовательности при помощи суперпозиции, примитивной рекурсии или минимизации.*

Доказательство. Следует из теорем этого параграфа и теоремы о Т-вычислимости рекурсивных функций. ∎

Теорема 4.39. *Следующие три класса арифметических функций совпадают:*

а) частично рекурсивные функции;

б) программно вычислимые функции;

в) вычислимые по Тьюрингу функции.

4.11 Универсальные функции

Нам далее потребуется константная функция τ_i от n аргументов, равная i при любых значениях своих аргументов. Эти функции для всех i являются рекурсивными и могут быть заданы как рекурсивные термы, например, следующим образом.

Пусть τ_0 есть $[\theta;\ \eta_1^n]$ и τ_{i+1} есть $[';\ \tau_i]$. Ясно, что τ_i равно i при любых значениях аргументов.

Определение 4.11.1. *Пусть K — класс n-местных арифметических функций. Функция ϕ от $n+1$ аргументов называется универсальной для K, если*

а) функция $[\phi;\ \tau_i,\ \eta_1^n,\dots,\eta_n^n]$ принадлежит к классу K при любом натуральном i;

б) для каждой функции ψ из K найдется такое натуральное i, что ψ есть $[\phi;\ \tau_i,\ \eta_1^n,\dots,\eta_n^n]$.

Теорема 4.40. *Не существует рекурсивной функции, универсальной для класса всех одноместных рекурсивных функций.*

Доказательство. Пусть ϕ — такая универсальная функция. Функция

$$[\overline{sg};\ [\phi;\ \eta_1^1,\ \eta_1^1]]$$

является одноместной рекурсивной. Значит, существует такое i, что

$$[\phi;\ \tau_i,\ \eta_1^1]$$

есть

$$[\overline{sg};\ [\phi;\ \eta_1^1,\ \eta_1^1]].$$

Однако,

$$[\phi;\ \tau_i,\ \eta_1^1](i)$$

есть $\phi(i,i)$, а

$$[\overline{sg};\ [\phi;\ \eta_1^1,\ \eta_1^1]](i)$$

есть $\overline{sg}(\phi(i,i))$. ∎

Теорема 4.41. *Для каждого натурального n существует частично рекурсивная функция, универсальная для класса n-местных частично рекурсивных функций.*

Доказательство. Будем использовать обозначения, введенные в доказательстве теоремы 4.37.

Функция Θ_{11} является универсальной для n-местных частично рекурсивных функций.

В самом деле, очевидно, что функция

$$[\Theta_{11};\ \tau_i,\ \eta_1^n, \ldots, \eta_n^n]$$

частично рекурсивна при любом натуральном i. С другой стороны, если ϕ является n-местной частично рекурсивной функцией, то она вычисляется некоторой МТ, пусть, номера m. Как показано в доказательстве теоремы 4.37, ϕ есть

$$[\Theta_{11};\ \tau_m,\ \eta_1^n, \ldots, \eta_n^n].$$

∎

Теорема 4.42. *Пусть $n = 1$ и Θ_{11} — введенная в доказательстве теоремы 4.37 функция, универсальная для одноместных частично рекурсивных. Не существует алгоритма для определения по паре натуральных чисел $\langle a, b\rangle$, определено ли $\Theta_{11}(a, b)$.*

Доказательство. Пусть существует такая рекурсивная функция f, что $f(a, b) = 1$, если $\Theta_{11}(a, b)$ определено, и $f(a, b) = 0$, если $\Theta_{11}(a, b)$ не определено. Будем считать, что $n = 1$ и использовать обозначения, введенные в предыдущем параграфе.

Пусть $\lambda_9(a, b)$ — это наименьшее из таких i, что

$$f(a, b) * exp(\Theta_8(a, \zeta_1(b), i), 1) = 0.$$

Если $\Theta_{11}(a, b)$ определено, то

$$\lambda_9(a, b) = \Theta_9(a, b).$$

Иначе, $\lambda_9(a, b) = 0$. Пусть

$$\lambda_{10}(a, b) = exp(\Theta_8(a, \zeta_1(b), \lambda_9(a, b)), 3).$$

Ясно, что

$$\lambda_{10}(a, b) = \Theta_{10}(a, b),$$

если $\Theta_{10}(a,b)$ определено. Иначе, $\lambda_{10}(a,b) = \psi_1(b)$.

Пусть, наконец,

$$\lambda_{11} = M[|-|;\ [exp;\ [\lambda_{10};\ \eta_1^3,\ \eta_2^3],\ \eta_3^3],\ [';[';[\theta;\ \eta_1^3]]]].$$

Ясно, что

$$\lambda_{11}(a,b) = \Theta_{11}(a,b),$$

если $\Theta_{11}(a,b)$ определено. Если ϕ — одноместная рекурсивная функция, то найдется такое a, что $\Theta_{11}(a,b) = \phi(b)$ для всех натуральных b. Тогда $\lambda_{11}(a,b) = \phi(b)$ для всех натуральных b. Поэтому, λ_{11} есть рекурсивная функция, универсальная для одноместных рекурсивных, что противоречит теореме 4.40. ∎

Теорема 4.43. *Существует одноместная частично рекурсивная функция ϕ', обладающая свойствами:*

а) не существует алгоритма для определения по натуральному числу a, определено ли $\phi'(a)$;

б) если $\phi'(a)$ определено, то $\phi'(a) = 0$.

Доказательство. Рассмотрим функцию

$$\phi = [\Theta_{11};\ [exp;\ \eta_1^1,\ \theta],\ [exp;\ \eta_1^1,\ [';\theta]]].$$

Ясно,что $\phi(a) = \Theta_{11}(exp(a,0), exp(a,1))$.

Пусть $f(a) = 1$, если $\phi(a)$ определено, и $f(a) = 0$, если $\phi(a)$ не определено. Пусть f — рекурсивная функция. Пусть $g(a,b) = f(2^a * 3^b)$. Пусть $\Theta_{11}(a,b)$ определено. Тогда $\phi(2^a * 3^b) = \Theta_{11}(a,b)$ тоже определено. Поэтому $f(2^a * 3^b) = 1$ и $g(a,b) = 1$. Если же $\Theta_{11}(a,b)$ не определено, то $\phi(2^a * 3^b)$ тоже не определено и $f(2^a * 3^b) = 0$. Значит, в этом случае $g(a,b) = 0$. Так как функция g — рекурсивна, то это противоречит теореме 4.42.

Ясно, что $[*;\ \phi,\ \theta]$ можно взять в качестве ϕ'. ∎

Упражнения к §§4.7—4.11

Упражнение 4.11.1. *Используя машины Тьюринга, существование которых утверждается в теореме 4.18, и операции над машинами Тьюринга, построить машины Тьюринга и доказать их корректность для вычисления следующих функций:*

а) $3 * (x!)$;

б) $2 + (x!)$;

в) целая часть кубического корня плюс один;

г) целая часть квадратного корня минус три, если это число неотрицательно, или нуль;

д) наибольшее из квадрата первого числа и корня квадратного из второго;

е) наименьшее из двух чисел;

ж) равной $x!$, *если* $x < y$, *и* $y!$ *в остальных случаях;*

з) равной 0, если $x < y$, *и* 1, *если* $x \geq y$;

и) равной $|x - y|$, *если* x — *четно, и* $2 * x$ *в остальных случаях;*

к) равной 2, если $x > 5$, *и* y *в остальных случаях;*

л) равной разности x *и* y, *если* $x > y$ *и* x — *простое число, нулю, если* $x \leq y$ *и* x — *простое число,* x *в остальных случаях;*

м) целая часть корня седьмой степени из произведения x *и* y;

н) равной x, *если* y — *простое число, и 2 в остальных случаях;*

о) $R(\eta_1^1,\ [*;\ \eta_3^3,\ \eta_1^3])$;

п) $R(\eta_1^1,\ [+;\ \eta_3^3,\ \eta_2^3])$;

р) $R(\eta_1^2,\ [*;\eta_3^4,\ \eta_4^4])$.

Упражнение 4.11.2. *Построить машины Тьюринга, моделирующие работу следующих программ, доказав это:*

а)

$i \leftarrow 0$; *если* $x = i$ *то* $x \leftarrow x$ *иначе*
если $y = i$ *то* $x \leftarrow x^2$ *иначе*
пока $x < y$ *делай* $y \leftarrow y;\ i \leftarrow i';\ x \leftarrow x'$ *все*
конец конец;

б)

$i \leftarrow 0$; *пока* $x < y$ *делай* $i \leftarrow i';\ x \leftarrow x'$ *все;*

в)

$t \leftarrow 0;\ i \leftarrow 0$; *пока* $i < x$ *делай*
$s \leftarrow y;\ j \leftarrow 0$; *пока* $j < y$ *делай*
$t \leftarrow t';\ t \leftarrow t';\ j \leftarrow j'$ *все;*
$i \leftarrow i';\ i \leftarrow i'$ *все;*

г)

$r \leftarrow 0;\ q \leftarrow x$; *пока* $y < q$ *делай*
$i \leftarrow 0;\ j \leftarrow y$; *пока* $j < q$ *делай*
$i \leftarrow i';\ j \leftarrow j'$ *все;*
$q \leftarrow i;\ r \leftarrow r'$ *все.*

Упражнение 4.11.3. *Для каждой из машин Тьюринга, построенных в пунктах 4), 5), 6), 8) доказательства теоремы 4.18, построить такую рекурсивную функцию* λ, *что если* n — *это номер незаключительного СП для этой МТ, то* $\lambda(n)$ — *это номер СП, в которое эта МТ* *перерабатывает СП номера n за один такт работы, и привести явное задание λ в виде рекурсивного терма (как, например, в упражнениях 4.6.1.о), 4.6.1.п), 4.6.1.р)).*

Предметный указатель

X-код, 337

автоматная грамматика, 73
аксиома ИВ, 105
аксиомы ИП, 142
алфавит, 11
арифметическая система, 343
асимптотически мажорируется, 275
асимптотической мажорантой, 275
ассоциативное исчисление, 237
атомная формула, 128

бескванторная формула, 154
булевы эквивалентности, 114

Высказывания, 100

верхняя оценка детерминированной (недетерминированной) временной (емкостной) сложности, 275
временная сложность, 274
вставимые нетерминалы, 58
вхождение высказывания в формулу, 101
вывод в КС-грамматике, 36
высота грамматики, 58
высота дерева, 96
высота символа, 58
высота языка, 58

глобальный предикат, 323
гомоморфизм графов, 66
грамматика без пустых правил, 44
грамматика с сильным самовставлением, 58
грамматикой с самовставлением, 58
граф, 62

дерево, 93
дерево вывода, 94
дерево доказательства, 107
деревом вывода слова, 95
детерминированный конечный автомат, 21
дизъюнктивная нормальная форма, 117
днф, 117
доказательство, 106, 143
дополнительные схемы правил вывода, 106

ёмкостная сложность, 274

замкнутость семейства языков, 29
запрос к базе данных, 323
значение терма на состоянии, 127
значение формулы, 102
значение формулы на состоянии, 130

индекс грамматики, 61
истинность секвенции, 104

Конкатенацией, 13

классы временной сложности, 285
классы емкостной сложности, 285
кнф, 117
код алгебраической системы, 337
код натурального числа, 336
код операции, 336

код предиката, 336
конечный автомат, 20
контекстно-свободная грамматика, 36
конфигурация 1-НМТ, 273
конфигурация МТ, 203
конъюнктивная нормальная форма, 117
критерий доказуемости кнф, 118
критерий доказуемости секвенции, 109
критерий доказуемости эд, 119
кс-грамматика, 36

леворекурсивная грамматика, 52
леворекурсивный символ, 52
лемма о замене в ИВ, 113
лемма о разрастании, 33
лемма об эквивалентности формул, 113
лента МТ, 203
линейная программа, 331

МП-автомат, 78

маркированной точкой, 309
машина Тьюринга, 200
машины Тьюринга с одной рабочей лентой, 270
минимизация, 179
модель множества формул, 158

Недетерминированная одноленточная машина машина Тьюринга, 272

непосредственная эквивалентность в ассоциативном исчислении, 237
непосредственное следствие слова в полусистеме Туэ, 232
нижняя оценка детерминированной (недетерминированной) временной (емкостной) сложности, 275
нормальная форма Грейбах, 55
нормальная форма Хомского, 44
нумерал, 250

объединение, 14
ограниченное перемножение, 181
ограниченное суммирование, 181
ответ на запрос, 323

Подсловом, 14

пересечения, 14
плотные классы сложности, 288
позиция в игре, 62
полнота множества формул, 158
полный класс глобальных предикатов, 342
полуканонический МП-автомат, 83
полусистема Туэ, 232
порядок вершин дерева, 94
правила вывода ИВ, 106
правила вывода ИП, 143
праволинейная грамматика, 73
предваренная формула, 154
префикса, 14
приведенная грамматика, 48
примитивная рекурсия, 178
присваивание, 171, 326
проблема соответствий Поста, 88
программа с метками, 188
простая программа, 326
простая программа со стеком, 343
простейшая функция, 180
противоречивое множества формул, 158
псевдоязык грамматики, 38

путевая грамматика, 332
путь в дереве, 96

работа конечного автомата, 21
работа МП-автомата, 79
работа МТ на СП, 201
работа МТ на ленте, 203
работа программы на состоянии, 172
работа программы с метками, 188
равенство термов, 128
разбор случаев, 182
разности, 14
распознавателем, 272, 274
распознается с детерминированной (недетерминированной) временной сложностью, 275
распознается с детерминированной (недетерминированной) емкостной сложностью, 275
регулярное выражение, 16
регулярность языка, 16
результат подстановки термов вместо переменных, 133
реконструкцией, 309
рекурсивная программа, 329

Сверхслово, 12
Словом, 11

самовставимый символ, 57
сводимые нетерминалы, 58
связь булевых функций и формул, 102
секвенциями, 103
сильная мажоранта, 276
сильно самовставимый символ, 58
символ секвенции, 103
следствие из множества формул, 158
следствие слова в полусистеме Туэ, 232
сложность глобального предиката, 341
сложность формулы логики предикатов, 130
соответствие Поста, 235
состояние, 127
состояние базы данных, 321
стековая машина, 301
структурированная программа, 171
суперпозиция арифметических функций, 177
суффиксом, 14
схема базы данных, 321
схема правил вывода от противного, 105
схемы главных правил вывода ИВ, 105
схемы правил вывода ИП, 142

тавтология, 103
теорема о доказуемости истинных секвенций, 120
теорема о кванторах, 143
теорема о композиции деревьев доказательства, 108
теорема о контекстно-свободности регулярных языков, 42
теорема о нормальной форме Грейбах, 55
теорема о нормальной форме Хомского, 47
теорема о полноте исчисления предикатов, 169
теорема о равенстве, 146
теорема о разрастании для КС-

языков, 96
теорема о совпадении класса регулярных языков и класса языков, задаваемых конечными автоматами, 73
теорема о существовании модели, 160
теорема о существовании днф, 118
теорема о существовании кнф, 117
теорема о существовании приведенной грамматики, 48
теорема о формальной индукции, 245
теорема об истинности доказуемых секвенций, 110
терм, 127
тест, 171
трасса, 68

универсальный язык относительно сводимости, 289
условие, 171

ФА, 244

формальная арифметика, 244
формула логики высказываний, 101
формула логики предикатов, 128
функциональная сложность формулы, 101
функция, вычислимая программой, 174

характеристическая функция, 175

частично рекурсивная функция, 180

эд, 117
эк, 117
эквивалентность автоматов, 22
эквивалентность в ассоциативном исчислении, 237
эквивалентность формул, 112, 150
эквивалентными, 272
элементарная дизъюнкция, 117
элементарная конъюнкция, 117

язык, 12
язык грамматики, 38
язык конечного автомата, 21