Оглавление

Глава 1

Математические модели алгоритмов

§ 1.1. Счетчиковые машины

§ 1.1.1. Определение

Определение 1 (Счетчиковая машина)**.** *Счетчиковой машиной (или машиной Минского) $\mathfrak{M}$ называется четверка (Q, n, P, q_0), где Q — непустое множество состояний, n — положительное натуральное число — количество счетчиков, P — программа счетчиковой машины, q_0 — элемент множества Q, начальное состояние.*

Программа P — это множество команд. Команда — слово одного из двух видов

$$q \to \mathrm{inc}_i, p$$

или

$$q \to \mathrm{dec}_i, p_1 : p_2,$$

где $q, p, p_1, p_2 \in Q$, i — натуральное число от 1 до n — номер счетчика.

Пример 1. *Рассмотрим пример счетчиковой машины. Пусть множество состояний Q состоит из шести элементов: $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$. Количество счетчиков будет 2, q_0 будет начальным состоянием. В качестве программы возьмем*

множество следующих команд:

$$q_0 \to \mathrm{dec}_2, q_1 : q_2$$
$$q_1 \to \mathrm{dec}_1, q_0 : q_3$$
$$q_3 \to \mathrm{inc}_1, q_4$$
$$q_4 \to \mathrm{dec}_2, q_3 : q_5$$

Определение 2 (Конфигурация счетчиковой машины)**.** *Конфигурация счетчиковой машины $\mathfrak{M} = (Q, n, P, q_0)$ — набор вида $(q, a_1, \ldots, a_n)$, где $q \in Q$ — состояние, n — количество счетчиков, $a_1, \ldots, a_n$ — натуральные числа. В этом случае q называется состоянием конфигурации, а числа $a_1, \ldots, a_n$ — значениями счетчиков $1, \ldots, n$ соответственно.*

Пример 2. *Для машины из примера 1 на предшествующей странице конфигурации могут быть, например, следующими: $(q_3, 3, 6)$ или $(q_5, 7, 10)$.*

Определение 3 (Начальная конфигурация)**.** *Если q_0 — начальное состояние, то конфигурация вида $(q_0, a_1, \ldots, a_n)$ называется начальной.*

Пример 3. *Начальными конфигурациями машины из примера 1 на предыдущей странице могут быть, например, $(q_0, 23, 6)$ или $(q_0, 0, 0)$.*

Определение 4 (Шаг работы счетчиковой машины)**.** *Счетчиковая машина $\mathfrak{M} = (Q, n, P, q_0)$ переводит за один шаг конфигурацию $\sigma = (q, a_1, \ldots, a_n)$ в конфигурацию $\tau = (p, b_1, \ldots, b_n)$, если выполняется одно из трех условий:*

1) *в программе P есть команда $q \to \mathrm{inc}_i, p$, $b_j = a_j$ при $j \neq i$, $b_i = a_i + 1$;*
2) *в программе P есть команда $q \to \mathrm{dec}_i, p : p'$, $b_j = a_j$ при $j \neq i$, $a_i > 0$ и $b_i = a_i - 1$;*
3) *в программе P есть команда $q \to \mathrm{dec}_i, p' : p$, $b_j = a_j$ при $j = 1, \ldots, n$ и $a_i = 0$.*

Обозначается: $\sigma \vdash {}^1_{\mathfrak{M}}\tau$ или просто $\sigma \vdash {}^1\tau$, если понятно о какой машине идет речь. Если $\sigma \vdash {}^1_{\mathfrak{M}}\tau$, то говорят также, что τ являются следующей за σ конфигурацией машины $\mathfrak{M}$.

Пример 4. *Рассмотрим счетчиковую машину* $\mathfrak{M}$ *из примера 1 на с. 3 и приведенные в качестве примеров конфигурации. Тогда*

$$(q_3, 3, 6) \vdash^1 (q_4, 4, 6);$$

$$(q_0, 23, 6) \vdash^1 (q_1, 23, 5);$$

$$(q_0, 0, 0) \vdash^1 (q_2, 0, 0).$$

Первое имеет место из-за команды $q_3 \to \mathrm{inc}_1, q_4$*, поэтому значение счетчика* 1 *увеличивается, значение счетчика* 2 *не меняется, состояние становится равным* q_4*. Второе — из-за команды* $q_0 \to \mathrm{dec}_2, q_1 : q_2$*, поскольку значение счетчика* 2 *положительно, то в следующей конфигурации его значение становится на один меньше, значение второго счетчика не меняется, а состояние становится* q_1*. Третье выполняется также из-за команды* $q_0 \to \mathrm{dec}_2, q_1 : q_2$*, но теперь значение счетчика* 2 *равно нулю, поэтому значения счетчиков остаются теми же, а состояние становится* q_2*.*

Для конфигурации $(q_5, 7, 10)$ *следующей нет, так как нет команд, начинающихся с* q_5*.*

Определение 5 (Детерминированная счетчиковая машина)**.** *Если в счетчиковой машине* $\mathfrak{M} = (Q, n, P, q_0)$ *для каждого* $q \in Q$ *существует не более одной команды, начинающейся с состояния* q*, то такая машина называется* д е т е р м и н и р о в а н н о й*.*

В дальнейшем мы будем рассматривать только детерминированные счетчиковые машины, не упоминая об этом специально. Машина из примера 1 на с. 3 является детерминированной.

Предложение 1. *Для детерминированной машины* $\mathfrak{M}$ *и конфигурации* σ *существует не более одной конфигурации* τ*, в которую* σ *переводится за один шаг.*

Доказательство. Если q — состояние конфигурации σ, то в детерминированной машине может быть не более одной команды, начинающейся с q.

Если такой команды нет, то следующей за σ конфигурации также не существует, поскольку не выполняется ни один из пунктов определения 4 на противоположной странице.

Если такая команда существует, то она является единственной. Но тогда выполняется в точности один из пунктов определения 4 на предыдущей странице, поэтому и следующая за σ конфигурация является единственной. □

Определение 6 (Заключительные состояние, конфигурация)**.** *Если в счетчиковой машине $\mathfrak{M} = (Q, n, P, q_0)$ для состояния $q \in Q$ программа P не содержит ни одной начинающейся с q команды, то такое состояние мы называем з а к л ю ч и т е л ь н ы м состоянием машины $\mathfrak{M}$. Конфигурацию с заключительным состоянием мы тоже называем з а к л ю ч и т е л ь н о й.*

Следствие 2. *Заключительная конфигурация не может переводится ни в какую конфигурацию за один шаг.*

Пример 5. *Для счетчиковой машины $\mathfrak{M}$ из примера 1 на с. 3 конфигурация $(q_5, 7, 10)$ является заключительной.*

Определение 7. *Пусть $\mathfrak{M} = (Q, n, P, q_0)$ — счетчиковая машина, σ и τ — ее конфигурации, k — положительное натуральное число. Говорят, что машина $\mathfrak{M}$ п е р е в о д и т з а k ш а г о в σ в τ (обозначается $\sigma \vdash^k_{\mathfrak{M}} \tau$), если существует конфигурация ρ такая, что $\sigma \vdash^1_{\mathfrak{M}} \rho$ и $\sigma \vdash^{k-1}_{\mathfrak{M}} \rho$, последнее считается определенным по индукции. Если $k = 0$, то $\sigma \vdash^k_{\mathfrak{M}} \tau$ тогда и только тогда, когда $\sigma = \tau$.*

Запись $\sigma \vdash^*_{\mathfrak{M}} \rho$ означает, что σ переводится в τ за какое-то количество шагов, то есть существует k для которого $\sigma \vdash^k_{\mathfrak{M}} \rho$.

Предложение 3. *Для детерминированной счетчиковой машины и конфигурации σ существует не более одной конфигурации τ, в которую σ переводится за k шагов.*

Доказательство. Используем индукцию по k — количеству шагов работы машины. Если $k = 0$, то по определению $\tau = \sigma$ и такое τ, естественно, является единственным.

Если $k = 1$, то утверждение повторяет предложение 1 на предыдущей странице.

Если $k > 1$ и существует конфигурация τ, для которой $\sigma \vdash^k_{\mathfrak{M}} \tau$, то по определению существует конфигурация ρ и выполняется $\sigma \vdash^1_{\mathfrak{M}} \rho$ и $\rho \vdash$ $\vdash^{k-1}_{\mathfrak{M}} \tau$. По предложению 1 на предшествующей странице такая конфигурация ρ является единственной. По индукционному предположению может существовать не более одного τ, для которого выполнено $\rho \vdash^{k-1}_{\mathfrak{M}} \tau$. □

Определение 8 (Остановка и зацикливание счетчиковой машины)**.** *Если для счетчиковой машины $\mathfrak{M}$ и конфигурации σ выполняется $\sigma \vdash$ $\vdash^k_{\mathfrak{M}} \tau$ и конфигурация τ является заключительной, то говорят, что машина о с т а н а в л и в а е т с я на конфигурации σ, а k называется*

количеством шагов (или временем) работы машины $\mathfrak{N}$ *на конфигурации* σ.

Если заключительной конфигурации τ, *для которой* $\sigma \vdash^*_{\mathfrak{M}} \tau$ *не существует, то говорят также, что машина* **зацикливается** *на конфигурации* σ.

Определение 9 (Вычисление функции на счетчиковой машине)**.** *Пусть* f — m-*местная частичная функция,* $\mathfrak{M} = (Q, n, P, q_0)$ — *счетчиковая машина и* $m \leqslant n$. *Машина* $\mathfrak{M}$ **вычисляет** f, *если для любой начальной конфигурации вида*

$$\sigma = (q_0, a_1, \ldots, a_m, 0, \ldots, 0)$$

выполнено одно из двух:

1) $f(a_1, \ldots, a_m) = b$ *и конфигурация* σ *переводится в некоторую заключительную конфигурацию вида* $(p, b, b_2, \ldots, b_n)$;

2) $f(a_1, \ldots, a_m)$ *неопределено и машина зацикливается на* σ.

Пример 6. *Рассмотрим односчетчиковую машину* $\mathfrak{N} = (\{q_0\}, 1, \varnothing, q_0)$ *с пустой программой. Поскольку в данном случае произвольная начальная конфигурация* (q_0, a) *является и заключительной, то результатом работы машины на конфигурации* (q_0, a) *само оно и будет. Следовательно, данная машина вычисляет одноместную тождественную функцию:* $e(a) = a$ *для всех* $a \in \omega$.

Отметим, что такая машина не вычисляет никакую нульместную функцию. В самом деле, чтобы машина $\mathfrak{N}$ *вычисляла какую-то нульместную функцию* $f()$ *по определению должно выполняться следующее: для любого* $a \in \omega$ *начальная конфигурация* (q_0, a) *должна переводится в конфигурацию вида* $(q, f())$, *где* q — *какое-то заключительное состояние. Следовательно, останавливаться машина должна на конфигурации, в которой значение счетчика* 1 *равно* $f()$, *то есть не зависит от начальной конфигурации. Но это не так.*

Пример 7. *Рассмотрим односчетчиковую машину* $\mathfrak{N} = (\{q_0, q_1\}, 1, P, q_0)$ *с программой, содержащей одну команду:* $q_0 \to \mathrm{inc}_1, q_1$. *В данном случае состояние* q_1 *будет заключительным. В данном примере начальная конфигурация* (q_0, a) *за один шаг перейдет в* $(q_1, a+1)$, *которая является заключительной. Значит, такая машина вычисляет одноместную функцию прибавления единицы:* $s(a) = a$ *для всех* $a \in \omega$.

Эта машина не вычисляет никакую нульместную функцию по тем же причинам, что и в предыдущем примере.

Пример 8. *Односчетчиковая машина* $\mathfrak{N} = (\{q_0\}, 1, P, q_0)$ *с однокомандной*

программой: $\{q_0 \to \mathrm{inc}_1, q_0\}$, *вычисляет нигде неопределенную функцию:* $f(a)$ *неопределено для любого* $a \in \omega$. *Очевидно, что заключительных конфигураций в данном случае не существует, так как нет заключительных состояний, поэтому машина на любой конфигурации зацикливается.*

Следствие 4. *Каждая счетчиковая машина* $\mathfrak{M} = (Q, n, P, q_0)$ *для каждого* $m = 1, \ldots, n$ *вычисляет некоторую* m*-местную частичную функцию.*

Определение 10 (Вычислимая функция)**.** *Если для (частичной) функции* f *существует счетчиковая машина, которая вычисляет* f*, то функцию* f *называют (алгоритмически) вычислимой.*

Пример 9. *Покажем, что существует двухсчетчиковая машина, вычисляющая сумму двух чисел. Пусть* $Q = \{q_0, q_1, q_2\}$, q_0 *— начальное состояние, программа* P *состоит из следующих команд:*

$$q_0 \to \mathrm{dec}_2, q_1 : q_2$$
$$q_1 \to \mathrm{inc}_1, q_0.$$

Рассмотрим машину $\mathfrak{M} = (Q, 2, P, q_0)$. *Ее начальная конфигурация имеет вид* (q_0, a, b), *где* a *и* b *— какие-то натуральные числа, начальные значения первого и второго счетчиков. Состояние* q_2 *у машины* $\mathfrak{M}$ *является заключительным, поскольку в программе* P *нет команд, начинающихся с* q_2.

Индукцией по b *покажем, что машина* $\mathfrak{M}$ *переводит любую конфигурацию вида* (q_0, a, b) *в заключительную конфигурацию* $(q_2, a + b, 0)$.

Если $b = 0$, *то конфигурация* $(q_0, a, 0)$ *из-за команды* $q_0 \to \mathrm{dec}_2, q_1 : q_2$ *по определению за один шаг перейдет в* $(q_2, a, 0)$, *что и требовалось.*

Предположим, что для b *данное утверждение доказано. Поскольку* $b + 1 > 0$, *то из-за команды* $q_0 \to \mathrm{dec}_2, q_1 : q_2$ *конфигурация* $(q_0, a, b+1)$ *перейдет в* (q_1, a, b). *В свою очередь из-за команды* $q_1 \to \mathrm{inc}_1, q_0$ *конфигурация* (q_1, a, b) *перейдет в* $(q_0, a + 1, b)$. *Поскольку для* b *наше утверждение доказано, то конфигурация* $(q_0, a + 1, b)$ *впоследствии перейдет в заключительную конфигурацию* $(q_0, (a + 1) + b, 0)$, *то есть* $(q_0, a + b + 1, 0)$.

Определение 11 (Ограниченная разность)**.** *Ограниченной разностью чисел* x *и* y *(обозначается с помощью* $\lfloor x - y \rfloor$*)называется следующая двухместная функция на множестве натуральных чисел:*

$$\lfloor x - y \rfloor = \begin{cases} x - y, \text{если } x - y \geqslant 0, \\ 0, \text{в противном случае.} \end{cases}$$

Пример 10. *Ограниченная разность является вычислимой функцией. Построим машину для ее вычисления:*

$$q_0 \to \mathrm{dec}_2, q_1 : q_2$$
$$q_1 \to \mathrm{dec}_1, q_0 : q_0.$$

Здесь q_0 — начальное, q_2 — заключительное состояние. Покажем, что каждая конфигурация (q_0, a, b) переводится в заключительную конфигурацию $(q_2, \lfloor a - b \rfloor, 0)$ индукцией по b.

Если $b = 0$, то из-за команды $q_0 \to \mathrm{dec}_2, q_1 : q_2$ следующей конфигурацией будет $(q_2, a, 0)$. Поскольку $b = 0$, то $\lfloor a - b \rfloor = a$, поэтому мы получили конфигурацию $(q_2, \lfloor a - b \rfloor, 0)$, что и требовалось.

Рассмотрим теперь случай $b+1$: конфигурацию $(q_0, a, b+1)$. При этом мы считаем, что для конфигураций вида (q_0, a, b) утверждение уже доказано. Очевидно, $b + 1 > 0$, поэтому получаем

$$(q_0, a, b + 1) \vdash {}^1(q_1, a, b).$$

Далее имеем

$$(q_1, a, b) \vdash {}^1(q_0, \lfloor a - 1 \rfloor, b).$$

По индукционному предположению получаем, что последняя конфигурация перейдет в $(q_0, \lfloor \lfloor a - 1 \rfloor - b \rfloor, 0)$.

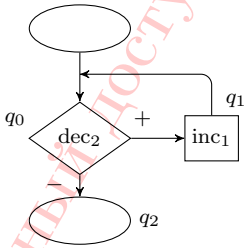
Возможны два случая. Если $a > 0$, то $\lfloor a - 1 \rfloor = a - 1$, поэтому $\lfloor \lfloor a - 1 \rfloor - b \rfloor = \lfloor a - 1 - b \rfloor = \lfloor a - (b + 1) \rfloor$, что и требуется.

Если $a = 0$, то $\lfloor a - 1 \rfloor = 0$, поэтому $\lfloor \lfloor a - 1 \rfloor - b \rfloor = \lfloor 0 - b \rfloor = 0 = \lfloor a - (b + 1) \rfloor$.

Счетчиковые машины удобно изображать в виде блок-схем. Каждое состояние соответствует вершине графа. Вершины, соответствующие состояниям q, для которых есть команда вида $q \to \mathrm{inc}_i, p$, изображаются в виде прямоугольника, из которого исходит одно ребро в вершину, соответствующую состоянию p. Вершины, соответствующие состояниям q, для которых есть команда вида $q \to \mathrm{dec}_i, p_1 : p_2$, изображаются в виде ромба, из которого исходят два ребра: помеченное знаком «+» ребро в вершину p_1, и помеченное знаком «−» ребро в вершину p_2. Вершины, соответствующие заключительным состояниям изображаются эллипсами. Чтобы отметить вершину, соответствующую начальному состоянию, добавляют еще одну вершину, тоже изображаемую эллипсом, из которой исходит одно ребро в начальную вершину. Отметим, что эту вершину нельзя спутать с вершиной, соответствующей заключительному состоянию, так как из последней не исходят ребра.

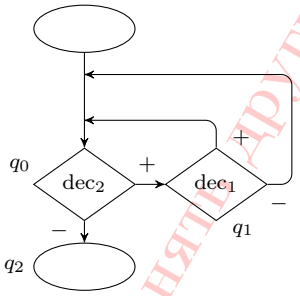
Обратное преобразование блок-схемы в счетчиковую машину также тривиально: вершины превращаются в состояния.

Пример 11. *Машину из примера 9 на с. 8 в виде блок-схемы можно изобразить следующим образом:*

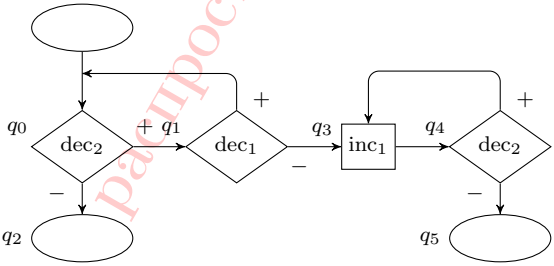


В данной схеме верхний эллипс не соответствует никакому состоянию и предназначен только для того, что отметить начальное — q_0. Ромб соответствует состоянию q_0 и команде $q_0 \to \mathrm{dec}_2, q_1 : q_2$, прямоугольник — состоянию q_1 и команде $q_1 \to \mathrm{inc}_1, q_0$, нижний эллипс — заключительному состоянию q_2.

Машина из примера 10 на предшествующей странице изображается так:



Самую первую из рассмотренных нами машин (пример 1 на с. 3) можно изобразить следующей блок-схемой:



Определение 12 (Эквивалентность счетчиковых машин)**.** *Пусть m — положительное натуральное число. Счетчиковые машины $\mathfrak{M}_1$ и $\mathfrak{M}_2$ называют m - э к в и в а л е н т н ы м и, если они вычисляют одну и ту же*

m-местную частичную функцию.

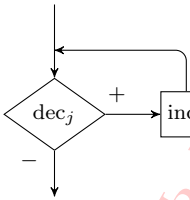
Следствие 5. *Если счетчиковые машины $\mathfrak{M}_1$ и $\mathfrak{M}_2$ являются m-эквивалентными, то они являются m' эквивалентными для любого $m' = 1, \ldots, m$.*

Следствие 6. *Для каждой счетчиковой машины $\mathfrak{M} = (Q, n, P, q_0)$ существует n-эквивалентная ей счетчиковая машина $\mathfrak{N} = (Q', n, P', q_0)$, имеющая в точности одно заключительное состояние.*

§ 1.1.2. Примеры счетчиковых машин

Предложение 7. *Существуют программа $\mathfrak{N}_{add(i,j)}$, которая добавляет к счетчику i значение счетчика j и обнуляет счетчик j, не изменяя остальных счетчиков.*

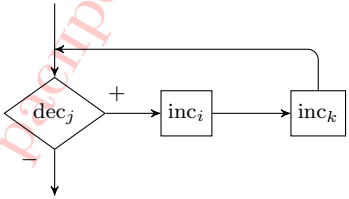
Доказательство. Достаточно модифицировать машину из примера 9 на с. 8 для работы со счетчиками i и j вместо 1 и 2 соответственно:



□

Следствие 8. *Существуют программа $\mathfrak{N}_{add(i,k,j)}$, которая добавляет к счетчикам i и k значение счетчика j и обнуляет счетчик j, не изменяя остальных счетчиков.*

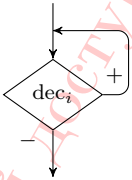
Доказательство. Достаточно вставить команду инкремента счетчика k:



□

Предложение 9. *Существуют программа $\mathfrak{N}_{zero(i)}$, обнуляющая счетчик i, не изменяя остальных счетчиков.*

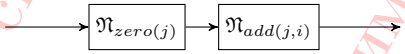
Доказательство.



□

Предложение 10. *Существует программа $\mathfrak{N}_{move(j,i)}$, перемещающая содержимое счетчика i в счетчик j и обнуляющая счетчик i, не изменяя остальных счетчиков.*

Доказательство. Нужно последовательно применить программы $\mathfrak{N}_{zero(j)}$ и $\mathfrak{N}_{add(j,i)}$:

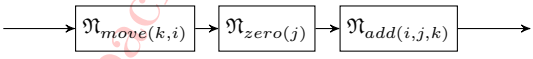


□

Для выполнения некоторых действий может потребоваться дополнительный счетчик, значение которого до и после выполнения программы счетчиковой машины для нас несущественно. Такие программы мы будем отмечать звездочкой: $\mathfrak{N}^*$.

Предложение 11. *Существует программа $\mathfrak{N}^*_{copy(j,i)}$, копирующая значение счетчика i в счетчик j, с использованием дополнительного счетчика k (значение счетчика i не меняется).*

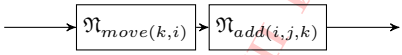
Доказательство. Данная программа состоит из таких частей: сначала содержимое i перемещаем во вспомогательный счетчик k при помощи $\mathfrak{N}_{move(k,i)}$ (одновременно обнуляется счетчик i), затем обнуляем счетчик j при помощи $\mathfrak{N}_{move(k,i)}$, наконец, добавляем значение счетчика k одновременно к счетчикам i и j с помощью $\mathfrak{N}_{add(i,j,k)}$:



□

Следствие 12. *Существует программа $\mathfrak{N}^*_{add(j,i)}$, добавляющая содержимое счетчика i к счетчику j, с использованием дополнительного счетчика k (значение счетчика i не меняется).*

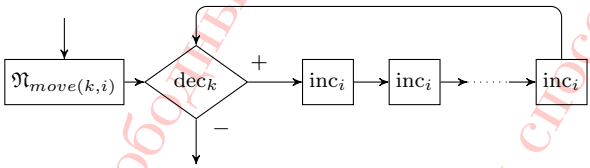
Доказательство. Достаточно в блок-схеме из доказательства утверждения 11 убрать часть, в которой обнуляется счетчик j:



□

Предложение 13. *Существует программа $\mathfrak{N}^*_{mult(i,c)}$, умножающая содержимое счетчика i на указанное натуральное число c, использующая дополнительно один счетчик k.*

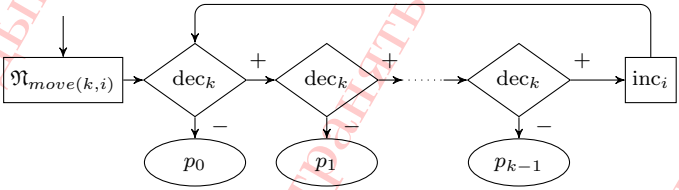
Доказательство.



Здесь справа расположено c команд инкремента. □

Предложение 14. *Существует программа $\mathfrak{N}^*_{div(i,c)}$, которая делит нацело содержимое счетчика i на указанное натуральное число c и переходит в одно из состояний $p_0, \ldots, p_{k-1}$ в зависимости от остатка. При этом используется один дополнительный счетчик k.*

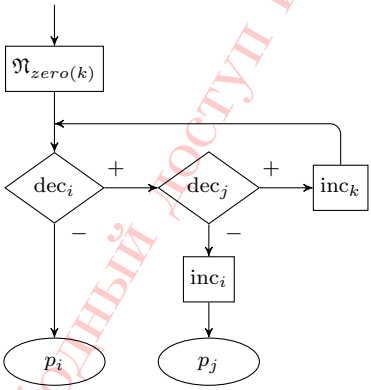
Доказательство.



□

Лемма 15. *Существует программа $\mathfrak{N}_{min(k,i,j)}$, которая помещает в счетчик k наименьшее из значений счетчиков i и j, вычитает из счетчиков i и j это значение и переходит в состояние p_i, если наименьшее значение содержалось в счетчике i, или в состояние p_j в противном случае.*

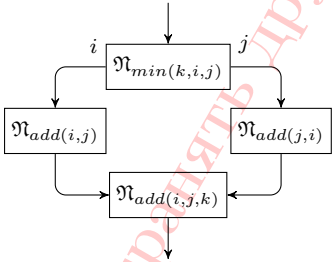
Доказательство.



□

Предложение 16. *Существует программа $\mathfrak{N}^*_{swap(i,j)}$, которая меняет значения счетчиков i и j местами с использованием одного дополнительного счетчика k.*

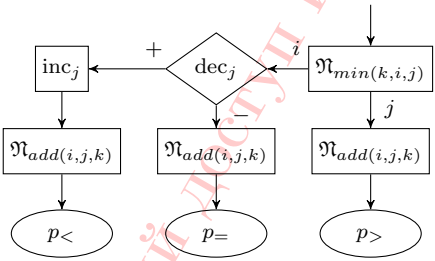
Доказательство.



□

Предложение 17. *Существует программа $\mathfrak{N}^*_{comp(i,j)}$, которая сравнивает содержимое счетчиков i и j и переходит в одно из состояний $p_<$, $p_=$ или $p_>$ в случае когда содержимое i-ого счетчика соответственно меньше, равно или больше чем содержимое j-го. При этом используется один дополнительный счетчик, а содержимое счетчиков i и j не меняется.*

Доказательство.



□

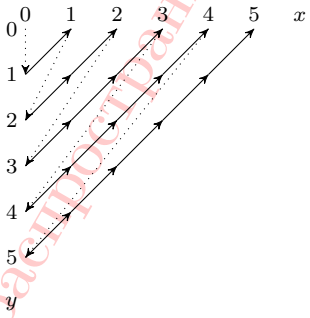
§ 1.1.3. Нумерация пар

Определение 13 (Канторова нумерация пар)**.** *С помощью $\langle x, y\rangle$ обозначим значение следующей двухместной функции*

$$\langle x, y\rangle = \frac{(x+y)(x+y+1)}{2} + x.$$

Данная функция называется канторовой нумерующей функцией.

Теорема 18. *Функция $\langle x, y\rangle$ взаимно однозначно отображает ω^2 на ω.*
Доказательство. Данное утверждение очевидно, если показать, в каком порядке нумеруются пары этой функцией:



Рассмотрим произвольную пару (x, y). Тогда количество пар, лежащих выше диагонали, на которой находится пара (x, y), равно

$$\frac{(x+y)(x+y+1)}{2}.$$

Добавив еще x (количество пар, лежащих на той же диагонали левее нашей) получим номер пары. □

Определение 14. *С помощью $\langle z\rangle_1$ и $\langle z\rangle_2$ обозначим обратные к $\langle x,y\rangle$ функции: $\langle\langle x,y\rangle\rangle_1 = x$, $\langle\langle x,y\rangle\rangle_2 = y$, $\langle\langle z\rangle_1, \langle z\rangle_2\rangle = z$.*

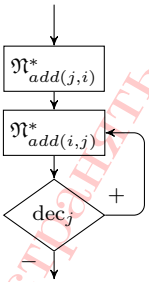
Предложение 19. *Существует счетчиковая машина $\mathfrak{N}^*_{pack(i,j)}$, которая записывает в счетчик i число $\langle x_i, x_j\rangle$, где x_i и x_j — значения счетчиков i и j соответственно. При этом используется один дополнительный счетчик.*

Доказательство. Сначала добавим значение счетчика i к счетчику j с помощью машины $\mathfrak{N}^*_{add(j,i)}$ из следствия 12 на с. 12. Далее будем добавлять к счетчику i элементы арифметической прогрессии:

$$(x_i + x_j) + (x_i + x_j - 1) + (x_i + x_j - 2) + \cdots + 1 + 0,$$

сумма которой как раз и будет равна

$$\frac{(x_i + x_j)(x_i + x_j + 1)}{2}.$$

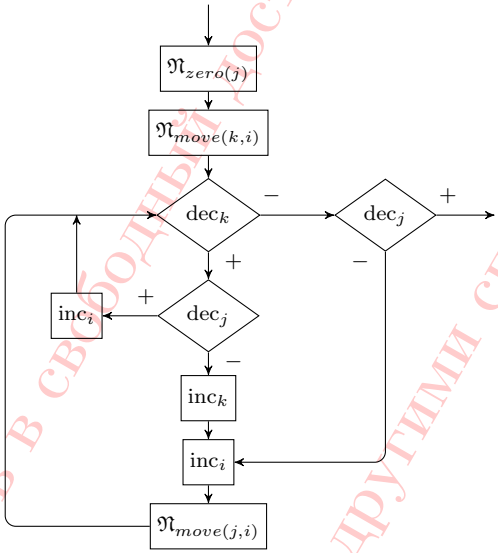


Дополнительный счетчик применяется как вспомогательный для машин $\mathfrak{N}^*_{add}$. □

Предложение 20. *Существует счетчиковая машина $\mathfrak{N}^*_{unpack(i,j)}$, которая записывает в счетчик i число $\langle z\rangle_1$, а в счетчик j — число $\langle z\rangle_2$, где где z — значение счетчика i. При этом используется один дополнительный счетчик k.*

Доказательство. Для решения данной задачи поступаем наоборот: из z (которое перемещено в счетчик k) последовательно вычитаем элементы арифметической прогрессии $0, 1, 2, 3, \ldots$, которые содержатся в счетчике j. Пусть $z = \langle x, y\rangle$. Тогда последний элемент, который можно вычесть — это $x + y$. После этого в счетчике k останется x, а счетчик j получит значение

$x + y + 1$. При следующем вычитании после выполнения x декрементов счетчик k обнулится, счетчик i будет содержать x, счетчик j — $y + 1$, а после декремента — y.



□

Определение 15 (Канторова нумерация n-ок)**.** *Распространим нумерацию пар на наборы произвольной длины:*

1) $\langle x \rangle = x$;

2) $\langle x_1, x_2 \rangle$ *определяется как раньше;*

3) $\langle x_1, \ldots, x_n \rangle = \langle \langle x_1, \ldots, x_{n-1} \rangle, x_n \rangle$, *здесь мы считаем, что* $\langle x_1, \ldots, x_{n-1} \rangle$ *определено по индукции. С помощью* $\langle z \rangle_n^m$ *будем обозначать функцию, возвращающую* m*-ый элемент* n*-ки, кодом которой является* z*, то есть обратные к нумерации* n*-ок функции.*

Предложение 21. *Для каждой вычислимой* m*-местной (частичной) функции* f *существует вычислимая одноместная (частичная) функция* f_1 *такая, что* $f(a_1, \ldots, a_m) = f_1(\langle a_1, \ldots, a_m \rangle)$.

Доказательство. Чтобы вычислить $f_1(z)$ достаточно проделать следующее: вычислить $\langle z \rangle_m^1, \ldots, \langle z \rangle_m^m$ и затем на полученных аргументах запустить машину, вычисляющую f. □

§ 1.1.4. Программы с метками и счетчиковые машины

Напомним, что (простой) программой с метками называется последовательность команд (операторов), каждая из которых имеет один из следующих видов:

1) $\alpha\ x = y;\ \beta$;

2) $\alpha\ x = s(y);\ \beta$;

3) $\alpha\ x = (y < z);\ \beta$;

4) α if x then β else γ.

Здесь α, β и γ — произвольные слова (имена меток), x, y и z — также слова (имена переменных). При этом у каждого оператора первая метка α не совпадает с первой меткой никакой другой команды. Первая метка первой команды программы называется начальной.

Конфигурацией программы с метками называется пара (λ, σ), где λ — метка программы, а σ — отображение множества переменных программы в предметную область (натуральные числа). Говорят, что конфигурация (α, σ) переводится программой Π за один шаг в конфигурацию (λ, τ) (обозначается $(\alpha, \sigma) \vdash \frac{1}{\Pi}(\lambda, \tau)$), если выполнено одно из условий:

1) в программе Π есть команда вида

$$\alpha\ x = y; \lambda,$$

$\tau(x) = \sigma(y)$ и $\tau(u) = \sigma(u)$ для переменных u отличных от x;

2) в программе Π есть команда вида

$$\alpha\ x = s(y); \lambda,$$

$\tau(x) = \sigma(y) + 1$ и $\tau(u) = \sigma(u)$ для переменных u отличных от x;

3) в программе Π есть команда вида

$$\alpha\ x = (y < z); \lambda,$$

$\tau(u) = \sigma(u)$ для переменных u отличных от x, $\tau(x) = 1$, если $\sigma(y) < \sigma(z)$, $\tau(x) = 0$ в противном случае;

4) в программе Π есть команда вида

$$\alpha \text{ if } x \text{ then } \lambda \text{ else } \gamma,$$

$\tau = \sigma$ и $\sigma(x) \neq 0$;

5) в программе Π есть команда вида

$$\alpha \text{ if } x \text{ then } \gamma \text{ else } \lambda,$$

$\tau = \sigma$ и $\sigma(x) = 0$.

Конфигурация (α, σ) переводится программой Π за m шагов в конфигурацию (λ, τ) (обозначается $(\alpha, \sigma) \vdash {}^{1}_{\Pi}(\lambda, \tau)$), если $(\alpha, \sigma) \vdash {}^{m-1}_{\Pi}(\mu, \rho)$ и $(\mu, \rho) \vdash {}^{1}_{\Pi}(\lambda, \tau)$ для некоторых ρ и μ. Запись $(\mu, \rho) \vdash {}^{*}_{\Pi}(\lambda, \tau)$ обозначает $(\mu, \rho) \vdash {}^{m}_{\Pi}(\lambda, \tau)$ для некоторого m.

Конфигурация (α, σ) называется заключительной, если в программе нет команды с меткой α. $\Pi(\alpha, \sigma) = (\beta, \tau)$, если $(\mu, \rho) \vdash {}^{*}_{\Pi}(\lambda, \tau)$ и (β, τ) является заключительной конфигурацией.

Говорим, что программа с метками Π вычисляет m-местную функцию f, если существуют попарно различные переменные $y_1, \ldots, y_m$ и переменная z (не обязательно отличная от $y_1, \ldots, y_m$) и для любой начальной конфигурации (α, σ), в которой $\sigma(y_i) = a_i$ для $i = 1, \ldots, m$, выполнено одно из двух:

1) $\Pi(\alpha, \sigma) = (\beta, \tau)$ и $\tau(z) = f(a_1, \ldots, a_m)$;

2) $\Pi(\alpha, \sigma)$ неопределено и $f(a_1, \ldots, a_m)$ неопределено.

Теорема 22. *Если m-местная (частичная) функция f вычислима на счетчиковой машине $\mathfrak{N} = (Q, n, P, q_0)$, то она вычисляется некоторой программой с метками.*

Доказательство. Именами переменных для программы с метками будут $x_1, \ldots, x_n$ и u, v. Метками программы с метками будут q, а также q^i, где $q \in Q$, $i = 1, \ldots, 6$. Для каждого состояния q счетчиковой машины определим один или два оператора $o(q)$ следующим образом:

1) если имеется команда вида $q \to \mathrm{inc}_i, p$, то $o(q)$ — это оператор

$$q\ x_i = s(x_i);\ p;$$

2) если имеется команда вида $q \to \mathrm{dec}_i, p_1 : p_2$, то $o(q)$ — это следующая последовательность операторов:

$$\left.\begin{array}{ll} q & \text{if } x_i \text{ then } q^1 \text{ else } p_2; \\ q^1 & u = 0;\ q^2; \\ q^2 & v = s(u);\ q^3; \\ q^3 & v = (v < x_i);\ q^4; \\ q^4 & \text{if } v \text{ then } q^5 \text{ else } q^6; \\ q^5 & u = s(u);\ q^2; \\ q^6 & x_i = u;\ p_1; \end{array}\right\} \tag{1.1}$$

3) если q — заключительное состояние, то $o(q)$ — пустая строка.

Рассмотрим программу с метками $o(q_0)o(q_1)\dots o(q_n)$, где q_0 — начальное состояние. Заметим, что начальной меткой программы Π является q_0.

Докажем следующее утверждение. Если $\sigma(x_i) = a_i$ для $i = 1, \dots, n$ и $(q, a_1, \dots, a_n) \vdash_{\mathfrak{N}}^{m} (p, b_1, \dots, b_n)$, то существует $M_m \geqslant m$ и $(q, \sigma) \vdash_{\Pi}^{M_m} (p, \tau)$ причем $\tau(x_i) = b_i$ для $i = 1, \dots, n$.

Применим индукцию по m — количеству шагов счетчиковой машины. Если $m = 0$, то утверждение очевидно, так как можно взять $\sigma = \tau$, $M_0 = 0$ и любая конфигурация переводится сама в себя за 0 шагов.

Пусть

$$(q, a_1, \dots, a_n) \vdash_{\mathfrak{N}}^{m-1} (r, c_1, \dots, c_n) \vdash_{\mathfrak{N}}^{1} (p, b_1, \dots, b_n).$$

По индукционному предположению $(q, \sigma) \vdash_{\Pi}^{M_{m-1}} (r, \rho)$ причем $\rho(x_i) = b_i$ для $i = 1, \dots, n$. Поскольку $(r, c_1, \dots, c_n) \vdash_{\mathfrak{N}}^{1} (p, b_1, \dots, b_n)$, то r не может быть заключительным состоянием. Дальше нужно рассмотреть три варианта.

1) В $\mathfrak{N}$ есть команда $r \to \mathrm{inc}_i, p$. Тогда $b_i = c_i + 1$, $b_j = c_j$ для $j \neq i$. По построению программы Π в ней присутствует оператор $r\ x_i = s(x_i);\ p$, поэтому согласно пункту 2) определения получаем $(r, \rho) \vdash_{\Pi}^{1} (p, \tau)$, причем $\tau(x_i) = \rho(x_i) + 1 = c_i + 1 = b_i$, $\tau(x_j) = \rho(x_j) = c_j = b_j$ при $j \neq i$. Получаем, $M_m = M_{m-1} + 1 \geqslant m - 1 + 1 \geqslant m$.

2) В $\mathfrak{N}$ есть команда $r \to \mathrm{dec}_i, p' : p$ и $c_i = 0$. Тогда $b_j = c_j$ для всех j. По построению программы Π в ней присутствуют операторы (1.1) Получаем, $M_m \geqslant M_{m-1} + 6 \geqslant m - 1 + 6 \geqslant m$.

3) В $\mathfrak{N}$ есть команда $r \to \mathrm{dec}_i, p' : p$ и $c_i = 0$. Тогда $b_j = c_j$ для всех j. По построению программы Π в ней присутствует оператор

$$r \text{ if } x_i \text{ then } r_1 \text{ else } p,$$

поэтому согласно пункту 5) определения получаем $(r, \rho) \vdash^{1}_{\Pi} (p, \tau)$, причем $\tau(x_j) = \rho(x_j) = c_j = b_j$ для всех j. Получаем, $M_m \geqslant M_{m-1} + 1 \geqslant m - 1 + 1 \geqslant m$.

Докажем теперь эквивалентность $\mathfrak{N}$ и Π. Пусть $\mathfrak{N}$ останавливается на начальной конфигурации $\sigma_2 = (q_0, a_1, \ldots, a_n)$ в конфигурации $\tau_2 = (q, b_1, \ldots, b_n)$. Это означает, что $\sigma_2 \vdash^{*}_{\mathfrak{N}} \tau_2$. Согласно утверждению $(q_0, \sigma_1) \vdash^{*}_{\Pi} (q, \tau_1)$. Поскольку q — заключительной состояние машины $\mathfrak{N}$, то в Π нет команды с меткой q, поэтому $\Pi(q_0, \sigma_1) = (q, \tau_1)$.

Пусть теперь $\mathfrak{N}$ зацикливается на конфигурации $\sigma_2 = (q_0, a_1, \ldots, a_n)$, но Π останавливается на (q_0, σ_1) за k шагов. Зацикливание $\mathfrak{N}$ значит, что если $\sigma_2 \vdash^{k}_{\mathfrak{N}} \tau_2$, то τ_2 — не заключительная конфигурация. Согласно утверждению $(q_0, \sigma_1) \vdash^{M_k}_{\Pi} (q, \tau_1)$ и q не может быть заключительной меткой. Но $M_k \geqslant k$, это противоречит тому, что Π за k шагов перешла в заключительную конфигурацию. Полученное противоречие показывает, что Π на (q_0, σ_1) также зацикливается. □

Теорема 23. *Если m-местная (частичная) функция f вычисляется программой с метками, то она вычисляется некоторой счетчиковой машиной.*

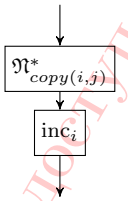
Доказательство. Будем считать, что имена переменных программы Π — это $x_1, \ldots, x_n$. Построим $n+1$-счетчиковую машину, обладающую следующим свойством: конфигурация $(q, a_1, \ldots, a_n, c)$ переводится в $(p, b_1, \ldots, b_n, d)$ для какого-то d тогда и только тогда конфигурация $(\{\}, q)$ программы Π переводится в $(\{\}, p)$. Счетчик с номером $n+1$ будет всегда использоваться в качестве вспомогательного.

Сначала по каждому оператору o программы Π построим часть блок-схемы $\mathfrak{N}_o$ счетчиковой машины.

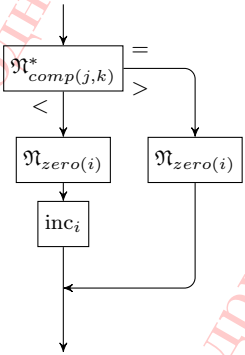
Для оператора o вида $\alpha\ x_i = 0;\ \beta$ в качестве $\mathfrak{N}_o$ берем $\mathfrak{N}_{zero(i)}$ из предложения 9 на с. 11.

Для оператора o вида $\alpha\ x_i = x_j;\ \beta$ в качестве $\mathfrak{N}_o$ берем $\mathfrak{N}_{copy(i,j)}$ из предложения 11 на с. 12.

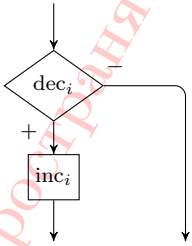
Для оператора o вида $\alpha\ x_i = s(x_j);\ \beta$ в качестве $\mathfrak{N}_o$ берем $\mathfrak{N}_{copy(i,j)}$ из предложения 11 на с. 12 за которой следует команда инкремента для счетчика i:



Для оператора o вида $\alpha\ x_i = (x_j < x_k);\ \beta$ строим $\mathfrak{N}_o$ с помощью $\mathfrak{N}_{comp}$ из предложения 17 на с. 14:



Для оператора o вида α if x_i then β else γ строим $\mathfrak{N}_o$ так:



□

§ 1.2. Машины Тьюринга

§ 1.2.1. Определения

Зафиксируем некоторый символ Λ. Будем считать, что он выбран раз и навсегда как некоторый фиксированный объект.

Определение 16 (Машина Тьюринга)**.** *Машиной Тьюринга* $\mathfrak{M}$ *называется четверка* (Q, Σ, P, q_0)*, где* Q *— множество состояний,* Σ *— алфавит ленты, причем* $Q \cap \Sigma = \varnothing$ *и* $\Lambda \in \Sigma$*,* P *— программа машины Тьюринга,* $q_0 \in Q$ *— начальное состояние.*

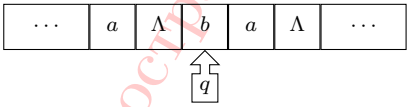
Программа машины Тьюринга — множество команд, каждая из которых имеет вид

$$q, a \to p, b, s,$$

где $q, p \in Q$ *— состояния,* $a, b \in \Sigma$ *— элементы алфавита (символы) ленты,* $s \in \{-1, 0, +1\}$ *— одно из трех чисел.*

Определение 17 (Конфигурация машины Тьюринга)**.** *Конфигурацией машины Тьюринга* $\mathfrak{M} = (Q, \Sigma, P, q_0)$ *называется тройка* (q, i, α)*, где* $q \in Q$ *— состояние конфигурации,* $i \in \mathbb{Z}$ *— целое число, называемое положением головки, а* α *— последовательность (бесконечная в обе стороны) символов* Σ*, которую называют лента. Символ* $\alpha(i)$ *называют обозреваемым символом. Иначе говоря,* α *— функция, отображающая множество целых чисел* $\mathbb{Z}$ *в* Σ*.*

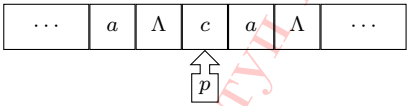
Пример 12. *Конфигурацию машины Тьюринга можно мыслить себе как разбитую на ячейки ленту, в каждой из этих ячеек записан один из символов алфавита* Σ*, а в одной из этих ячеек располагается рабочая головка, находящаяся в некотором состоянии из множества* Q*:*



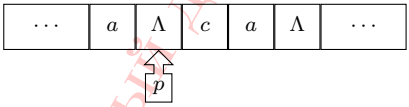
Здесь графически проиллюстрирована конфигурация (q, i, α)*, в которой* $\alpha(i-2) = a$*,* $\alpha(i-1) = \Lambda$*,* $\alpha(i) = b$*,* $\alpha(i+1) = a$*,* $\alpha(i+2) = \Lambda$*.*

Определение 18 (Шаг работы)**.** *Конфигурация* $\sigma = (q, i, \alpha)$ *переводится за один шаг машиной Тьюринга* $\mathfrak{M} = (Q, \Sigma, P, q_0)$ *в конфигурацию* $\tau = (p, j, \beta)$ *если в программе* P *есть команда* $q, a \to p, b, s$*,* $j = i + s$*,* $\beta(k) = \alpha(k)$ *при* $k \neq i$ *и* $\beta(i) = b$*.*

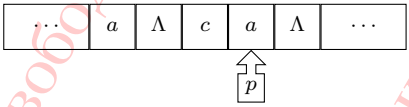
Пример 13. *Если в программе машины* $\mathfrak{M}$ *есть команда* $q, b \to p, c, 0$*, то конфигурация из примера 12 за один шаг может перейти в конфигурацию*



При наличии команды $q, b \to p, c, -1$, та же конфигурация может перейти в



а при наличии команды $q, b \to p, c, +1$ — в



Определение 19 (Заключительная конфигурация)**.** *Конфигурация (q, i, α) машины Тьюринга $\mathfrak{M} = (Q, \Sigma, P, q_0)$ называется заключительной, если в программе P нет ни одной команды вида $q, \alpha(i) \to p, b, s$.*

Следствие 24. *Заключительная конфигурация не переводится ни в какую другую за один шаг.*

Следующее определение почти дословно повторяет аналогичное для счетчиковых машин.

Определение 20. *Пусть $\mathfrak{M} = (Q, \Sigma, P, q_0)$ — машина Тьюринга, σ и τ — ее конфигурации, k — положительное натуральное число. Говорят, что машина $\mathfrak{M}$ переводит за k шагов σ в τ (обозначается $\sigma \vdash^k_{\mathfrak{M}} \tau$), если существует конфигурация ρ такая, что $\sigma \vdash^1_{\mathfrak{M}} \rho$ и $\sigma \vdash^{k-1}_{\mathfrak{M}} \rho$, последнее считается определенным по индукции. Если $k = 0$, то $\sigma \vdash^k_{\mathfrak{M}} \tau$ тогда и только тогда, когда $\sigma = \tau$.*

Запись $\sigma \vdash^*_{\mathfrak{M}} \rho$ означает, что σ переводится в τ за какое-то количество шагов, то есть существует k для которого $\sigma \vdash^k_{\mathfrak{M}} \rho$.

Определение 21 (Остановка, зацикливание)**.** *Машина Тьюринга $\mathfrak{M}$ останавливается за k шагов на конфигурации σ, если $\sigma \vdash^k_{\mathfrak{M}} \tau$ для некоторой заключительной конфигурации τ. Машина $\mathfrak{M}$ останавливается на σ, если она останавливается за некоторое количество шагов.*

В противном случае говорят, что машина зацикливается на

конфигурации σ.

Определение 22 (Детерминированная машина Тьюринга)**.** *Машина Тьбюринга называется детерминированной, если для каждого состояния $q \in Q$ в программе P существует не более одной команды вида $q, a \to p, b, s$.*

Доказательство следующего утверждения выполняется так же как для аналогичного предложения 3 на с. 6 для счетчиковых машин:

Предложение 25. *Для детерминированной машины Тьюринга и конфигурации σ существует не более одной конфигурации τ, в которую σ переводится за k шагов.*

Определение 23 (Слово, записанное на ленте)**.** *Предположим, что в конфигурации $\sigma = (q, i, \alpha)$ для всех $k \in \mathbb{Z}$, кроме конечного их числа выполнено $\alpha(k) = \Lambda$. Пусть k_1 — наименьшее из чисел, для которых $\alpha(k_1) \neq \Lambda$, k_2 — наибольшее из них. Тогда говорят, что в конфигурации α на ленте записано слово $\alpha(k_1)\alpha(k_1+)\ldots\alpha(k_2)$, начиная с ячейки k_1.*

Если выполнено $\alpha(k) = \Lambda$ для всех k, то говорят, что на ленте записано пустое слово.

Определение 24 (Начальная конфигурация)**.** *Начальной конфигурацией машины Тьюринга $\mathfrak{M} = (Q, \Sigma, P, q_0)$ называется конфигурация вида $(q_0, 0, \alpha)$, причем $\alpha(k) = \Lambda$ для всех $k < 0$ и для всех $k \geqslant M$, где M — некоторое натуральное число.*

Таким образом, в начальной конфигурации все ячейки, кроме конечного их числа содержат Λ. Слово, записанное на ленте в начальной конфигурации, называется входным словом.

Определение 25 (Результат работы машины Тьюринга)**.** *Пусть $\mathfrak{M} = (Q, \Sigma, P, q_0)$ — машина Тьюринга. Слово v называется результатом работы $\mathfrak{M}$ на слове u и обозначается с помощью $\mathfrak{M}(u)$, если начальная конфигурация $\sigma = (q_0, 0, \alpha)$, в которой на ленте написано слово u, начиная с ячейки 0, переводится в (заключительную) конфигурацию τ, в которой на ленте написано слово v. Если $\sigma \vdash_{\mathfrak{M}}^{k} \tau$, то k называется количеством шагов (или временем) работы машины $\mathfrak{M}$ на слове u.*

Если на конфигурации $(q_0, 0, \alpha)$ машина $\mathfrak{M}$ зацикливается, то говорят, что результат $\mathfrak{M}(u)$ неопределен.

Определение 26 (Эквивалентность машин Тьюринга)**.** *Машины Тьюринга $\mathfrak{M}_1$ и $\mathfrak{M}_2$ называются эквивалентными, если для любого слова u результаты $\mathfrak{M}_1(u)$ и $\mathfrak{M}_2(u)$ или одновременно определены и равны, или одновременно неопределены.*

Определение 27 (Рабочее пространство)**.** *Пусть σ — начальная конфигурация машины Тьюринга $\mathfrak{M}$ для входного слова u. Рабочим пространством $\mathfrak{M}$ на u называется множество номеров ячеек i таких, что $\sigma^*_{\mathfrak{M}}(q, i, \alpha)$ для некоторых q и $\mathfrak{M}$.*

Неформально, рабочее пространство — множество ячеек, в которых головка машины побывала хотя бы раз при работе на слове u.

Определение 28 (Нормальный вход)**.** *Входное слово u называется нормальным для машины $\mathfrak{M}$, если оно находится в начальной конфигурации внутри рабочего пространства.*

Таким образом, вход называется нормальным, если машина Тьюринга просматривает его целиком.

Определение 29 (Стандартная заключительная конфигурация)**.** *Заключительная конфигурация (q, i, α) называется стандартной, если $i = 0$ (головка находится в нулевой ячейке) и написанное на ленте слово начинается в нулевой ячейке (если оно непусто).*

Машина Тьюринга $\mathfrak{M}$ имеет стандартную заключительную конфигурацию, если любая начальная конфигурация σ с нормальным входом переводится машиной $\mathfrak{M}$ в некоторую стандартную заключительную, если только $\mathfrak{M}$ на σ останавливается.

Предложение 26. *Для каждой машины Тьюринга $\mathfrak{M} = (Q, \Sigma, P, q_0)$ существует эквивалентная ей машина Тьюринга $\mathfrak{M}'$ со стандартной заключительной конфигурацией.*

Доказательство. Алфавит Σ' новой машины будет состоять из всех старых символов Σ, а также новых символов, получаемых из старых добавлением одного или двух штрихов (мы предполагаем, что старый алфавит

Σ таких символов не содержит):

$$\Sigma' = \Sigma \cup \{a' : a \in \Sigma\} \cup \{a'' : a \in \Sigma\}.$$

Штрихованые символы будут записываться в ячейки, которые машина посетила, дважды штрихованые — помещаться в ячейку с нулевым номером. Чтобы этого добиться преобразуем программу P. Введем новое начальное состояние q_0' и добавим всевозможные команды вида $q_0', a \to q_0, a'', 0$ для всех символов $a \in \Sigma$. Каждую команду вида $q, a \to p, b, s$ заменяем на три: $q, a \to p, b', s$, $q, a' \to p, b', s$ и $q, a'' \to p, b'', s$.

После того, как старая машина перейдет в заключительное состояние, новая должна будет проделать дополнительные действия. Нужно найти на «заштрихованной» части ленты слово-результат, переместить его так, чтобы начало находилось в ячейке, помеченной двумя штрихами, и, наконец, убрать все штрихи (начиная с правого конца результата, чтобы последней командой «очистить» от двух штрихов нулевую ячейку). □

§ 1.2.2. Кодирование чисел и слов

Счетчиковые машины и машины Тьюринга имеют одно существенное отличие: счетчиковые машины изначально работают с натуральными числами, а машины Тьюринга — со словами. Чтобы иметь возможность сравнивать их необходимо научиться представлять одно другим.

Определение 30 (Унарный код)**.** *Пусть зафиксирован некоторый символ a. Унарным кодом числа n называется слово, состоящее из $n+1$ символа a.*

Наиболее популярным символом для унарного кодирования является палочка «|».

Пример 14. *Число пять кодируется в унарной форме словом «||||||», а число нуль — словом «|».*

Заметим, что унарный код является симметричным: неважно с какой стороны начинать запись слева или справа.

Определение 31 (Позиционный код)**.** *Пусть p — натуральное число не меньшее 2, и зафиксированы некоторые символы $a_0, \ldots, a_{p-1}$. p-ичным кодом числа n называется слово $a_{i_k} a_{i_{k-1}} \ldots a_{i_1} a_{i_0}$, $k > 0$, причем*

1) $n = \sum_{j=0}^{k} i_j \times p^j$;

2) если $k > 0$, то $i_k \neq 0$.

Число k называется p-ичной разрядностью числа n.

Чаще всего мы будем применять для позиционной записи чисел обычные цифры: $0, 1, 2, \ldots$ Наиболее распространено двоичное кодирование.

Пример 15. *Двоичный код числа нуль — это слово «0», а числа пять — «101».*

Позиционный код в отличие от унарного симметричным не является, при записи кода слева направо получиться вообще говоря не то слово, что и при записи справа налево. Обычно, мы будем придерживаться традиционной формы записи чисел: старшие разряды располагаются слева.

Определение 32. *Пусть зафиксирован некоторый способ кодирования чисел словами (унарная или p-ичная система). Пусть f — m-местная частичная функция, $\mathfrak{M}$ — машина Тьюринга. Машина $\mathfrak{M}$ вычисляет f, если для любой начальной конфигурации σ, в которой записанное слово на ленте имеет вид $w_1 \Lambda w_2 \Lambda \ldots \Lambda w_m$, где w_i — код числа a_i, выполнено одно из двух:*

1) *$f(a_1, \ldots, a_m) = b$ и конфигурация σ переводится в некоторую заключительную конфигурацию, в которой на ленте записан код числа b;*

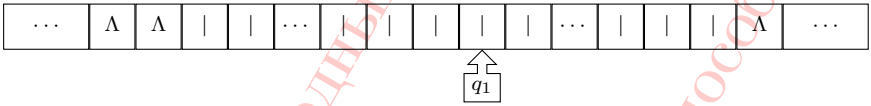
2) *$f(a_1, \ldots, a_m)$ неопределено и машина зацикливается на σ.*

Пример 16. *Построим машину Тьюринга, вычисляющую сумму двух чисел в унарной системе. Пусть для унарного кодирования применяется символ «|».*

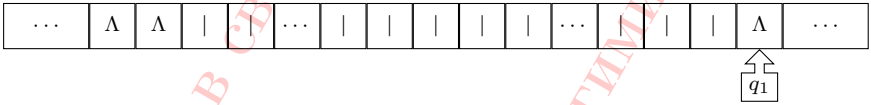
$$
\begin{array}{l}
q_0, | \ \to q_0, \ |, +1 \\
q_0, \Lambda \to q_1, \ |, +1 \\
q_1, | \ \to q_1, \ |, +1 \\
q_1, \Lambda \to q_2, \Lambda, -1 \\
q_2, | \ \to q_3, \Lambda, -1 \\
q_3, | \ \to q_4, \Lambda, -1 \\
q_4, | \ \to q_4, \ |, -1 \\
q_4, \Lambda \to q_5, \Lambda, +1
\end{array}
$$

Чтобы сложить два натуральных числа a и b с использованием этой машины, мы должны запустить ее в конфигурации

Здесь первая группа содержит $a+1$ палочку, а вторая — $b+1$. Благодаря первым трем командам, головка начнет двигаться вправо, оставаясь в состоянии q_0, заменит первый пустой символ на палочку, перейдет при этом в состояние q_1:

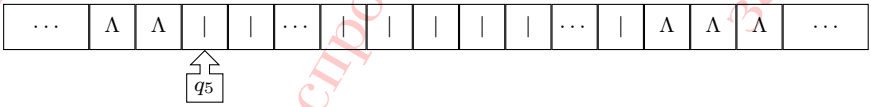


и продолжит движение вправо, пока конфигурация не станет такой:



Здесь образовалась едина группа из $a+b+3$ палочек. Чтобы получить правильный унарный код суммы $a+b$ нужно стереть две последних палочки:

Находясь в состоянии q_4 головка будет двигаться влево, пока обозреваемым символом не станет пустой, после чего вернется к начальной ячейке и закончит свою работу в конфигурации:



Заметим, что в отличие от счетчиковых машин не каждая машина Тьюринга вычисляет некоторую частичную функцию, поскольку заключительная конфигурация может содержать какое-то слово, не являющееся кодом никакого числа.

Рассмотрим противоположную задачу — кодирование слов числами.

Пусть алфавит Σ конечен и содержит $p-1$ символов. Тогда мы можем пронумеровать его элементы: $\Sigma = \{a_1, \ldots, a_{p-1}\}$, и рассматривать любое

слово в алфавите Σ как запись некоторого натурального числа в p-ичной системе счисления. При этом символ a_i рассматривается как цифра i. Такое кодирование допускает обратное преобразование — каждому числу n соответствует не более одного слова, кодом которого число n является.

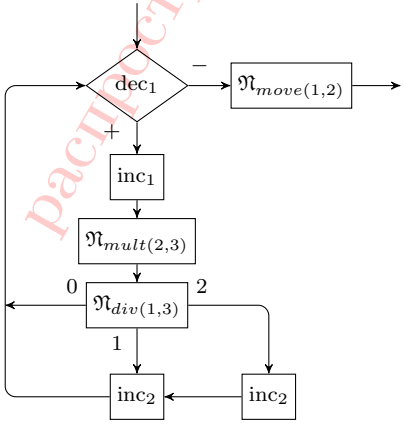
В общем случае при алфавите из p символом кодирование в p-ичной системе невозможно. В этом случае один из символов, например, a, должен соответствовать цифре 0. Но тогда слова w и aw получат один и тот же код, поскольку нули в крайних левых разрядах роли не играют. Если разряды рассматриваются в обратном порядке (слева располагаются младшие), то по аналогии можно рассмотреть слова w и wa.

Однако на практике нам часто будут встречаться ситуации, когда пустой символ в начале или конце слова не должен рассматриваться. Пример — определение 23 на с. 25 записанного на ленте слова. Тогда мы можем отождествить пустой символ с цифрой 0, что не повредит кодированию.

Пример 17. *Рассмотрим, как будет выглядеть задача «переворачивания» слова, сформулированная в «числовом» виде. Будем по прежнему рассматривать слова в двухсимвольном алфавите $\{a, b\}$. Поскольку в слове каждая из букв может стоять на любом месте, в том числе — в начале и конце, то двоичную систему счисления для кодирования слов в данном случае использовать невозможно. Поэтому мы будем считать, что каждое слово — это запись некоторого числа в т р о и ч н о й системе, причем символ a рассматривается как цифра 1, а b — как цифра 2. Например, слово abbab будет кодироваться числом*

$$1 \times 3^4 + 2 \times 3^3 + 2 \times 3^2 + 1 \times 3 + 2 = 160.$$

Задача «переворачивания» слова, таким образом, превращается в задачу нахождения числа, троичная запись которого получается «переворачиванием» троичной записи аргумента. Машина, решающая такую задачу, может выглядеть так:



Отметим, что для чисел, троичная запись которых содержит цифру 0*, данная операция в общем не является обратимой, так как нули, которые стояли на младших разрядах, окажутся на старших и будут потеряны.*

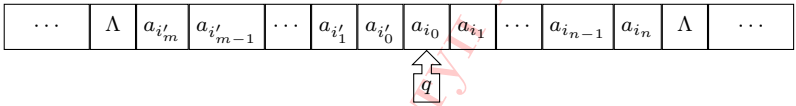
В некоторых случаях нам придется рассматривать слова, в которых может встречаться сколь угодно много различных символов, то есть слова в бесконечном алфавите. Чтобы указать способ кодирования слов числами в такой ситуации, пронумеруем символы алфавита натуральными числами: $\Sigma = \{a_0, a_1, a_2, \dots\}$. Рассмотрим двухсимвольный алфавит $\Sigma_2 = \{1, 0\}$. Каждому символу $a_i \in \Sigma$ сопоставим слово в алфавите Σ_2: $c(a_i) = 10^i$. Тогда в каждом слове $w = w_1 \dots w_n$ в алфавите Σ мы можем выполнить замену каждого символа w_j на $c(w_j)$: $c(w) = c(w_1) \dots c(w_n)$. Такая замена взаимно однозначна, так как количество нулей, стоящих после очередной единицы однозначно определяет символ, стоящий на этом месте. Далее полученное слово можно рассмотреть как запись числа в двоичной системе и получить таким образом его код.

§ 1.2.3. Моделирование машины Тьюринга на счетчиковой машине

Теорема 27. *Для каждой машины Тьюринга $\mathfrak{M}$, вычисляющей одноместную (частичную) функцию f, существует счетчиковая машина $\mathfrak{N}$ с тремя счетчиками, вычисляющая ту же (частичную) функцию f, что и $\mathfrak{M}$.*

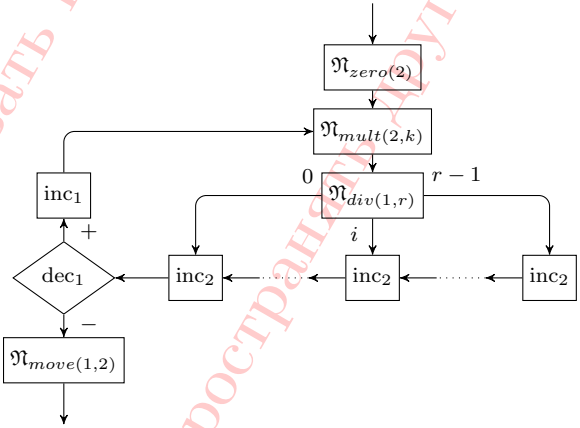
Доказательство. Работа счетчиковой машины будет состоять из трех этапов. Первый — подготовка к моделированию машины Тьюринга, второй — само моделирование, третий — раскодирование результата.

Основная идея моделирования состоит в том, что в счетчике 1 будет хранится содержимое ленты справа от головки (обозреваемый символ будет тоже входить в правую часть), в счетчике 2 — слева, а счетчик x_3 будет применяться как вспомогательный. Пусть алфавит машины $\mathfrak{M}$ состоит из k символов: $a_0, \dots, a_{k-1}$, при этом мы полагаем, что a_0 — пустой символ. Содержимое ленты $a_{i_1} a_{i_2} \dots a_{i_m}$ мы кодируем числом, запись которого в p-ичной системе выглядит как $i_m i_{m-1} \dots i_2 i_1$. При кодировании содержимого ленты младшие разряды соответствуют ячейкам, находящимся ближе к головке. При таком кодировании пустые символы в начале и конце ленты будут соответствовать старшим нулевым разрядам в позиционной записи числа. Таким образом, для конфигурации

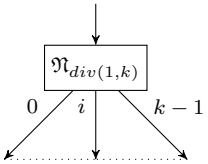


в счетчике 1 будет записано число $\sum_{j=0}^{n} i_j \times k^j$ а в счетчике 2 — число $\sum_{j=0}^{m} i'_j \times k^j$.

В начальной конфигурации все (непустое) содержимое ленты находится справа от головки, а левая часть ленты пуста. Поэтому первая часть (подготовка к моделированию) будет заключаться в подготовке кода ленты в счетчике 1 и обнулении счетчика 2. Предположим, для записи чисел машина Тьюринга использует r-ичную систему счисления с цифрами «0», ..., «$r-1$», при этом цифры пишутся в обычном порядке: справа младшие разряды, слева — старшие. Напомним, нулем мы кодируем пустой символ, поэтому будем считать, что коды цифр «0», ..., «$r-1$» равны 1, ..., r соответственно. Для преобразования применим счетчиковую машину:

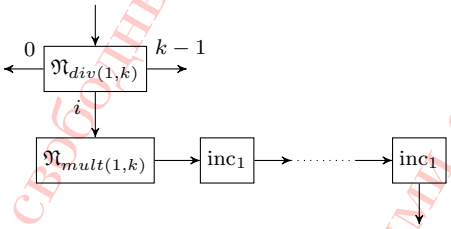


Для каждого из состояний q машины $\mathfrak{M}$ выберем все команды, которые с него начинаются и построим фрагмент программы счетчиковой машины, моделирующей работу $\mathfrak{M}$ в q. Начинаться этот фрагмент будет с запуска программы $\mathfrak{N}_{div(1,k)}$ (предложение 14 на с. 13):



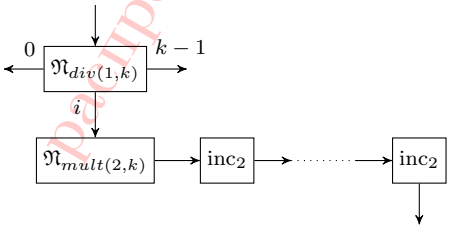
Таким образом, мы сможем определить, какой символ находится в ячейке, где располагается головка машины. Затем, в зависимости от команды, соответствующей состоянию q и символу, выполняем одну из следующих программ.

Команды машины могут быть трех видов. Первый вариант: $q, a_i \to p, a_j, 0$, то есть обозреваемая ячейка не меняется. Чтобы промоделировать такую команду на счетчиковой машине продолжим программу следующим образом:



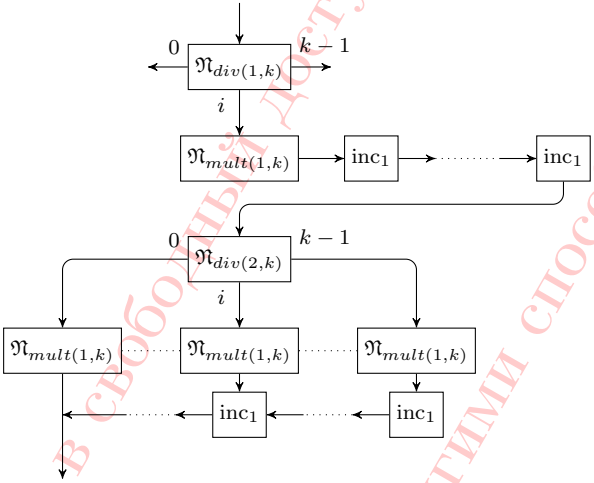
Здесь сначала счетчик 1 делится на k и находится остаток от деления (то есть определяется, на какой символ смотрим головка) с помощью программы $\mathfrak{N}_{div(1,k)}$ из предложения 14 на с. 13. Затем, если в результате выполнения $\mathfrak{N}_{div(1,k)}$ было определено, что остаток от деления равен i, то счетчик 1 умножается на k с помощью программы $\mathfrak{N}_{mult(1,k)}$ из предложения 13 на с. 13, и, наконец, счетчик 1 увеличивается еще на j единиц (записать в текущую ячейку символ a_j), после чего переходим в состояние p. Заметим, что при выполнении команды $q, a_i \to p, a_j, 0$ левая часть ленты не меняется, как и счетчик 2.

Второй вариант команды: $q, a_i \to p, a_j, +1$, то есть головка машины сдвигается вправо. Продолжение программы в этом случае выглядит похоже:



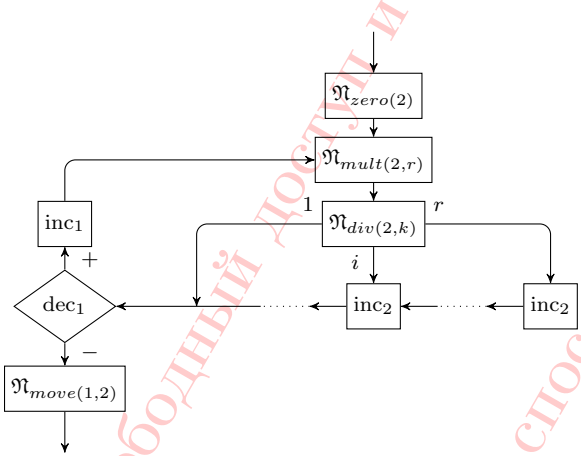
Отличие только в том, что после деления увеличивается счетчик 2, так как после выполнения команды, та ячейка, куда записано a_j будет располагаться левее головки.

Третий случай: $q, a_i \to p, a_j, -1$, головка машины сдвигается влево. Начало такое же как в первом случае:



Но после того, как записан новый символ нужно промоделировать перемещение головки влево, то есть младший разряд числа из счетчика 2 нужно переписать в счетчик 1. Для этого мы при помощи программы $\mathfrak{N}_{div(2,k)}$ определяем, какой символ стоял слева от головки, после чего переносим его вправо, умножим счетчик 1 на k и увеличив его столько раз, чему был равен остаток при последнем делении.

Последняя часть работы счетчиковой машины — «раскодирование» ленты машины Тьюринга. Напомним, что корректная запись числа в r-ичной системе состоит цифр «0», ..., «$r-1$», кодируемых с помощью $1, \ldots, r$ соответственно. Поэтому никаких других остатков от деления быть не может. Для этого используем машину:



□

Следствие 28. *Пусть $m \geqslant 1$ и m-местная (частичная) функция f вычислима на машине Тьюринга. Тогда f вычислима на счетчиковой машине с $m + 2$ счетчиками.*

Доказательство. Отличие будет заключаться только в первой части доказательства: после построения кода для первого аргумента нужно будет добавить коды второго и следующего аргументов, разделяя их пустым символом □

Следствие 29. *Пусть $m \geqslant 3$ и m-местная (частичная) функция f вычислима на машине Тьюринга. Тогда f вычислима на счетчиковой машине с $m + 1$ счетчиком.*

Доказательство. С помощью машины, вычисляющей канторову нумерацию, можно записать номер пары аргументов, например, a_{m-1} и a_m, в один из этих счетчиков, например, $m - 1$. В результате счетчик m освободится и может быть использован вместе со счетчиком $m + 1$ как вспомогательный для построения кода первых $m - 2$ аргументов. После этого из счетчика $m-1$ извлекаются два последних аргумента и по очереди добавляются к уже построенному коду. □

§ 1.2.4. Односторонние машины Тьюринга

Зафиксируем кроме пустого символа еще один стандартный символ — #.

Определение 33. *Машина Тьюринга $\mathfrak{M} = (Q, \Sigma, P, q_0)$ называется односторонней, если $\Sigma \ni \#$ и ее программа P не содержит команд ни одного из следующих видов:*

1) $q, \# \to p, b, -1$;

2) $q, \# \to p, b, s$, *где* $b \neq \#$;

3) $q, a \to p, \#, s$, *где* $a \neq \#$.

Таким образом, односторонняя машина не может сдвинуть головку левее символа #, не может вместо # записать какой-то другой символ и не может записать # вместо какого-то другого символа.

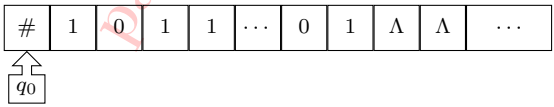
Определение 34. *Начальной конфигурацией односторонней машины Тьюринга $\mathfrak{M} = (Q, \Sigma, P, q_0)$ называется начальная конфигурация $(q_0, 0, \alpha)$, в которой $\alpha(0) = \#$ и $\alpha(k) \neq \#$ для всех остальных k.*

Следовательно, с помощью # отмечена начальная ячейка, в которой находится головка, и эту метку нельзя ни уничтожить ни скопировать, также как и сдвинуться левее ее.

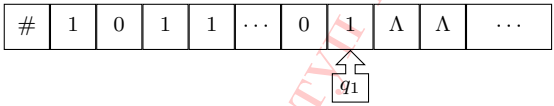
Пример 18. *Рассмотрим одностороннюю машину Тьюринга, вычисляющую одноместную функцию прибавления единицы в двоичной системе. Будем считать, что для записи чисел используются обычные цифры* 0 *и* 1.

$$
\begin{array}{l}
q_0, 1 \to q_0, 1, +1 \\
q_0, 0 \to q_0, 0, +1 \\
q_0, \Lambda \to q_1, \Lambda, -1 \\
q_1, 1 \to q_1^1, 0, -1 \\
q_1, 0 \to q_3, 1, -1 \\
q_1^1, 1 \to q_1^1, 0, -1 \\
q_1^1, 0 \to q_3, 1, -1
\end{array}
\qquad
\begin{array}{l}
q_1^1, \# \to q_2^1, \#, +1 \\
q_2^1, 0 \to q_2, 1, +1 \\
q_2, 0 \to q_2, 0, +1 \\
q_2, \Lambda \to q_3, 0, -1 \\
q_3, 1 \to q_3, 1, -1 \\
q_3, 0 \to q_3, 0, -1
\end{array}
$$

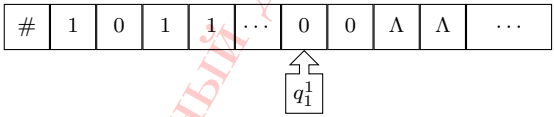
Начальная конфигурация для данной задачи может выглядеть так:



Головка будет двигаться вправо до пустого символа после чего разворачивается:



Состояние q_1^1 используется для обозначения переноса из предыдущего разряда:



Как только появится возможность перенос будет записан на место нуля, после чего в состоянии q_3 головка будет возвращена к началу ленты. Если же нулевых разрядов в числе нет, то новая единица будет записана в первую ячейку, а все остальные — сдвинуты на одну позицию вправо (там могут быть только нули).

Теорема 30. *Для каждой счетчиковой машины $\mathfrak{N} = (Q, n, P, q_0)$, вычисляющей m-местную (частичную) функцию f, и любого натурального положительного k существует односторонняя машина Тьюринга $\mathfrak{M} = (U, \Sigma, R, u_0)$ с $k+1$-символьным алфавитом Σ, вычисляющая ту же (частичную) функцию f с использованием k-ичной записи (если $k = 1$ — унарной).*

Доказательство. Основная идея моделирования вычисления счетчиковой машины на машине Тьюринга заключается в следующем: конфигурации $(q, a_1, \ldots, a_n)$ счетчиковой машины $\mathfrak{N}$ будет соответствовать слово на ленте

$$\#w_1\Lambda w_2\Lambda \ldots \Lambda w_n,$$

где w_i — k-ичная запись числа a_i. Напомним, что k-ичная запись числа всегда непуста (для унарного представления количество единиц на один больше кодируемого числа).

В начальный момент конфигурация счетчиковой машины выглядит следующим образом: $(q_0, a_1, \ldots, a_m, a_{m+1}, \ldots, a_n)$. Здесь $a_1, \ldots, a_m$ — аргументы функции, а $a_{m+1}, \ldots, a_n$ — произвольные числа. При вычислении функции f на машине Тьюринга, в начальной конфигурации на ленте будет записано слово

$$\#w_1\Lambda w_2\Lambda \ldots \Lambda w_m,$$

которое, возможно, отличается от нужного нам, количеством чисел. Поэтому первое, что будет делать машина Тьюринга — добавлять значения счетчиков, в которых не передаются аргументы. Поскольку в начале вычисления их значения могут быть произвольными, то можно считать их нулями.

Рассмотрим более подробно реализацию работу машины Тьюринга с унарным представлением счетчиков. Остальные варианты будут отличаться только деталями увеличения/уменьшения счетчика. Для унарного представления дописывание значений дополнительных счетчиков можно проделать при помощи следующей программы:

$$
\begin{aligned}
u_0, | &\to u_0, |, +1;\\
u_0, \Lambda &\to u_1, \Lambda, +1;\\
u_1, | &\to u_0, |, +1;\\
u_1, \Lambda &\to u'_1, |, +1;\\
u'_1, \Lambda &\to u_2, \Lambda, +1;\\
u_2, \Lambda &\to u'_1, |, +1;\\
u'_2, \Lambda &\to u_2, \Lambda, +1;\\
&\ldots,\\
u_{n-m}, \Lambda &\to v, |, -1;\\
v, \Lambda &\to v, \Lambda, -1;\\
v, | &\to v, |, -1;\\
v, \# &\to q_0, \#, 0.
\end{aligned}
$$

Состояние u_0 является начальным состоянием машины $\mathfrak{M}$. Данный фрагмент, начиная с того места, где стоят два пустых символа подряд, дописывает $n - m$ палочек, разделяя их пустым символом.

Мы будем строить нашу машину Тьюринга так, что после моделирования каждой команды счетчиковой машины головка возвращается к начальной ячейке #. Таким образом, выполнение очередной команды счетчиковой машины начинается в конфигурации вида $(q, 0, \alpha)$.

Предположим, команда имеет вид $q \to \mathrm{inc}_i, p$. Тогда моделирующий фрагмент программы машины Тьюринга $\mathfrak{M}$ выглядит так:

$$
\begin{aligned}
q, \# &\to q_1, \#, +1\\
q_1, | &\to q_1, |, +1\\
q_1, \Lambda &\to q_2, \Lambda, +1\\
q_2, | &\to q_2, |, +1\\
q_2, \Lambda &\to q_3, \Lambda, +1\\
&\ldots,\\
q_{i-1}, \Lambda &\to q_i, \Lambda, +1\\
q_i, | &\to q_i, |, +1
\end{aligned}
\qquad
\begin{aligned}
q_i, \Lambda &\to q^{\Lambda}, |, +1\\
q^{\Lambda}, | &\to q^{|}, \Lambda, +1\\
q^{|}, | &\to q^1, |, +1\\
q^{|}, \Lambda &\to q^{\Lambda}, |, +1\\
q^{\Lambda}, \Lambda &\to q^2, \Lambda, -1\\
q^2, \Lambda &\to q^2, \Lambda, -1\\
q^2, | &\to q^2, |, -1\\
q^2, \# &\to p, \#, 0
\end{aligned}
$$

То есть мы доходим до i-го счетчика (состояния q_1–q_i), добавляем к нему единицу (состояние q_i), а содержимое ленты справа от него сдвигаем на

одну ячейку вправо (состояния $q^|$ и q^Λ, верхний индекс означает очередной переносимый символ). Два подряд идущих пустых символа указывают на окончание процедуры, после чего, как мы уже упоминали, возвращаем головку в начальную позицию (состояние q^2).

Для команды вида $q \to \mathrm{dec}_i, p : p'$ моделирование будет таким:

$$
\begin{array}{rcl}
q, \# & \to & q_1, \#, +1 \\
q_1, | & \to & q_1, |, +1 \\
q_1, \Lambda & \to & q_2, \Lambda, +1 \\
q_2, | & \to & q_2, |, +1 \\
q_2, \Lambda & \to & q_3, \Lambda, +1 \\
\dots, & & \\
q_{i-1}, \Lambda & \to & q_i, \Lambda, +1 \\
q_i, | & \to & q'_i, |, +1 \\
q'_i, | & \to & q^|, |, -1 \\
q'_i, \Lambda & \to & q^2, \Lambda, -1 \\
q^2, \Lambda & \to & q^2, \Lambda, -1 \\
q^2, | & \to & q^2, |, -1 \\
q^2, \# & \to & p', \#, -1
\end{array}
\qquad
\begin{array}{rcl}
q^|, | & \to & q_f, |, +1 \\
q^|, \Lambda & \to & q_f, |, +1 \\
q^\Lambda, | & \to & q_f, \Lambda, +1 \\
q^\Lambda, \Lambda & \to & q_f, \Lambda, +1 \\
q_f, | & \to & q_{f|}, |, +1 \\
q_f, \Lambda & \to & q_{f\Lambda}, |, +1 \\
q_{f|}, | & \to & q^|, |, -1 \\
q_{f|}, \Lambda & \to & q^\Lambda, \Lambda, -1 \\
q_{f\Lambda}, | & \to & q^|, |, -1 \\
q_{f\Lambda}, \Lambda & \to & q^3, \Lambda, -1 \\
q^3, \Lambda & \to & q^3, \Lambda, -1 \\
q^3, | & \to & q^3, |, -1 \\
q^3, \# & \to & p, \#, -1
\end{array}
$$

Здесь мы доходим до i-го счетчика (состояния q_1–q_i), проверяем, содержит ли он больше одной единицы (состояние q'_i), если нет — возвращаемся назад (состояние q^2) и переходим в состояние p'. В противном случае мы содержимое ленты справа от текущей ячейки сдвигаем на одну позицию влево по одному символу, при этом одна палочка исчезает. Дойдя до двух пустых символов, мы возвращаемся к началу (состояние q^3) и переходим в состояние p. □

Следствие 31. *Для каждой вычислимой функции существует вычисляющая ее односторонняя машина Тьюринга с двухсимвольным алфавитом.*

Следствие 32. *Для каждой вычислимой одноместной функции f существует вычисляющая ее счетчиковая машина с тремя счетчиками.*

Доказательство. Для функции f существует вычисляющая ее счетчиковая машина $\mathfrak{N}$. Согласно теореме 30 на с. 37 для $\mathfrak{N}$ существует машина Тьюринга $\mathfrak{M}$, вычисляющая ту же самую функцию. Согласно теореме 27 на с. 31 для $\mathfrak{M}$ найдется трехсчетчиковая машина $\mathfrak{N}$, вычисляющая ту же саму функцию. □

§ 1.2.5. Количество состояний

Теорема 33. *Для каждой машины Тьюринга* $\mathfrak{M} = (Q, \Sigma, P, q_0)$ *существует эквивалентная машина Тьюринга* $\mathfrak{N} = (Q', \Sigma', P', u_0)$ *с 4 состояниями:* $Q' = \{u_0, u_1, u_2, u_3\}$.

Доказательство. Будем считать, что состояния машины Тьюринга $\mathfrak{M}$ пронумерованы натуральными числами: $Q = \{q_0, \ldots, q_h\}$.

Расширим множество входных символов следующим образом. Для каждого символа $a \in \Sigma$ и натурального $x \leqslant h$ включаем в Σ' следующие символы:

$$a, (a, x), (a, x, +1, t), (a, x, +1, f), (a, x, -1, t), (a, x, -1, f).$$

Интуитивно, символ (a, x) для машины $\mathfrak{N}$ будет означать, что для машины $\mathfrak{M}$ в данной ячейке с символом a находится головка в состоянии q_x. Четыре других новых символа будут применяться, чтобы промоделировать перемещение головки в нужном состоянии в соседнюю ячейку.

Новая машина будет иметь два «рабочих» состояния: u_1 и u_2, начальное состояние u_0 и состояние u_3, предназначенное для завершения работы.

Для u_0 в машине $\mathfrak{N}$ будут только команды, которые предназначены для «запуска» моделирования: $u_0, a \to u_1, (a, 0), 0$, для всевозможных $a \in \Sigma$.

Проще всего промоделировать выполнение команды машины $\mathfrak{M}$ вида $q_i, a \to q_j, b, 0$. В новой машине $\mathfrak{N}$ ей соответствует одна команда: $u_1, (a, i) \to u_1, (b, j), 0$.

Для моделирования команды машины $\mathfrak{M}$ вида $q_i, a \to q_j, b, +1$ в программу машины $\mathfrak{N}$ вставим следующее. Сначала запомним в текущей ячейке новое состояние:

$$u_1, (a, i) \to u_1, (b, j, +1, f), +1. \tag{1.2}$$

В следующей ячейке подготовимся к «приему» информации из предыдущей:

$$u_1, c \to u_1, (c, 0, -1, t), -1. \tag{1.3}$$

Затем по единице перемещаем номер состояния из старой ячейки в новую:

$$\begin{array}{l} u_1, (b, x, +1, f) \to u_1, (b, x-1, +1, f), +1; \\ u_1, (c, y, -1, t) \to u_1, (c, y+1, -1, t), -1 \end{array} \tag{1.4}$$

для $x > 0$. Наконец, когда «перемещение» закончилось, записываем в обе ячейки «обычное» содержимое:

$$\begin{array}{l} u_1, (b, 0, +1, f) \to u_2, \quad b \quad, +1; \\ u_2, (c, y, -1, t) \to u_1, (c, y), 0. \end{array} \tag{1.5}$$

Таким образом, третий элемент «составного» символа означает в какую сторону следует перемещаться из данной ячейки, а четвертый — является ли ячейка старой f или новой t.

По аналогии в машине $\mathfrak{N}$ моделируется команда машины $\mathfrak{M}$ вида $q_i, a \to q_j, b, -1$:

$$\begin{array}{lll}
u_1, (a, i) & \to u_2, & (b, j, -1, f) \quad , -1; \\
u_2, c & \to u_1, & (c, 0, +1, t) \quad , +1; \\
u_1, (b, x, -1, f) & \to u_1, & (b, x-1, -1, f), -1; \\
u_1, (c, y, +1, t) & \to u_1, & (c, y+1, +1, t), +1; \\
u_1, (b, 0, -1, f) & \to u_2, & b \quad , -1; \\
u_2, (c, y, +1, t) & \to u_1, & (c, y) \quad , 0.
\end{array}$$

Наконец, если в программе машины $\mathfrak{M}$ нет команды вида $q_i, a \to \ldots$, то в программу машины $\mathfrak{N}$ вставляем завершающий фрагмент: $u_1, (a, i) \to u_3, a, 0$.

Для доказательства корректности построения нужно доказать такое утверждение: если для машины $\mathfrak{M}$ выполняется

$$\sigma = (q_i, n, \alpha) \vdash^k *_{\mathfrak{M}} (q_j, m, \beta) = \tau$$

для некоторого k, то для машины $\mathfrak{N}$ выполняется

$$\sigma'(u_1, n, \alpha') \vdash_{\mathfrak{N}}^{K_k} (u_1, m, \beta') = \tau'$$

для некоторого $K_k \geqslant k$. Здесь α' и β' совпадают с α и β соответственно, за исключением того, что $\alpha'(n) = (\alpha(n), i)$ и $\beta'(m) = (\beta(m), j)$.

Доказывается утверждение индукцией по k — количеству шагов работы машины $\mathfrak{M}$. Если $k = 0$, то обе конфигурации машины $\mathfrak{M}$ совпадают. Но тогда обе конфигурации машины $\mathfrak{N}$ тоже совпадают и σ' переводится в τ' за нуль шагов, то есть $K_0 = 0$.

Предположим, что для $k-1$ утверждение доказано. Тогда

$$(q_i, n, \alpha) \vdash_{\mathfrak{M}}^{k-1} (q_l, r, \gamma) \vdash_{\mathfrak{M}}^{1} (q_j, m, \beta).$$

По индукционному предположению

$$(u_1, n, \alpha') \vdash_{\mathfrak{M}}^{K_{k-1}} (u_1, r, \gamma'),$$

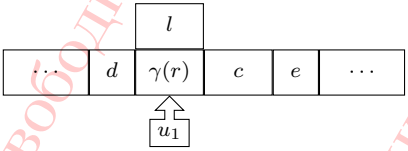
где $K_{k-1} \geqslant k-1$, γ' совпадает с γ, за исключением того, что $\gamma'(r) = (\gamma(r), l)$. Далее нужно рассмотреть варианты, какая команда машины $\mathfrak{M}$ применялась при переходе от конфигурации (q_l, r, γ) к (q_j, m, β).

Если переход был выполнен с помощью команды $q_l, \gamma(r) \to q_j, \beta(n), 0$, то $r = m$. Но тогда в программе машины $\mathfrak{N}$ есть команда $u_1, (\gamma(r), l) \to u_1, (\beta(m), j), 0$, поэтому

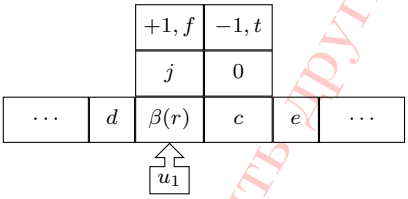
$$(u_1, r, \gamma') \vdash_{\mathfrak{M}}^{1} (u_1, m, \beta'),$$

следовательно, $K_k = K_{k-1} + 1 \geqslant k - 1 + 1 = k$.

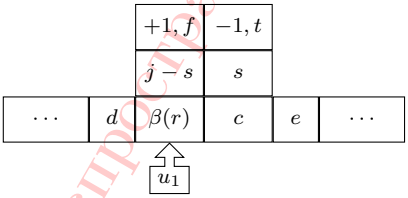
Если переход был выполнен с помощью команды $q_l, \gamma(r) \to q_j, \beta(r), +1$, то $m = r + 1$. Но тогда в программе машины $\mathfrak{N}$ есть команды (1.2)–(1.5). При выполнении первых двух из них из конфигурации



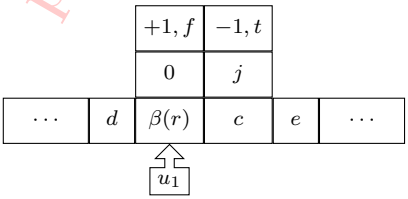
мы получим конфигурацию:



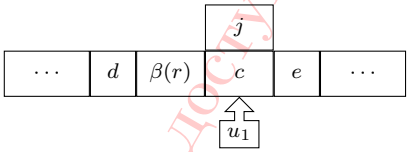
Затем, после s выполнений пары команд (1.4) мы получим



Следовательно, после j выполнений будет конфигурация



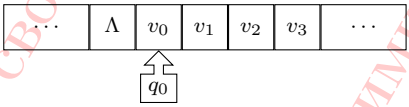
Затем, выполнив команды (1.5), мы получим



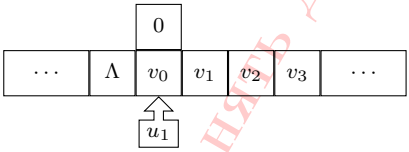
то есть (u_1, n, β'), что и требовалось. Таким образом, $K_k = K_{k-1} + 2 + 2 \times j + 2 \geqslant k - 1 + 4 \times (j + 1) \geqslant k$.

Аналогично рассматривается случай с командой $q_l, \gamma(r) \to q_j, \beta(r), -1$.

Теперь легко доказать эквивалентность $\mathfrak{M}$ и $\mathfrak{N}$. Пусть $\mathfrak{M}(v)$ определено и σ начальная конфигурация машины $\mathfrak{M}$:



Рассмотрим начальную конфигурацию машины $\mathfrak{N}$ для того слова. После выполнения команды $u_0, v_0 \to u_1, (v_0, 0), 0$ получаем конфигурацию σ':



Согласно утверждению, поскольку $\mathfrak{M}$ на конфигурации σ переходит в некоторую заключительную конфигурацию $\tau = (q_f, m, \beta)$, то $\mathfrak{N}$ на конфигурации σ' перейдет в конфигурацию $\tau' = (u_1, m, \beta')$, где β' отличается от β только тем, что $\beta'(m) = (\beta(m), f)$. Поскольку для машины $\mathfrak{M}$ конфигурация τ является заключительной, то в программе $\mathfrak{M}$ нет команды вида $q_f, \beta(m) \to \cdots$. Следовательно, в программе машины $\mathfrak{N}$ есть команда $u_1, (\beta(m), f) \to u_3, \beta(m), 0$, поэтому после τ' машина $\mathfrak{N}$ перейдет в конфигурацию (u_3, m, β), которая будет заключительной и в которой на ленте написано то же самое слово. Таким образом мы получили $\mathfrak{M}(v) = \mathfrak{N}(v)$.

Предположим теперь, что $\mathfrak{M}(v)$ зацикливается. Допустим, $\mathfrak{N}(v)$ останавливается. Аналогично предыдущему случаю после выполнения первой команды машина $\mathfrak{N}$ окажется в конфигурации σ'. Пусть она переходит в некоторую заключительную конфигурацию за K шагов. Поскольку $\mathfrak{M}$ при работе на σ зацикливается, то за K шагов σ будет переведено в некоторую

незаключительную конфигурацию τ. Согласно утверждению, машина $\mathfrak{N}$ не менее чем за K шагов перейдет в конфигурацию τ', которая тоже не будет заключительной. Противоречие, следовательно, такого K существовать не может и машина $\mathfrak{N}$ при работе на слове v также зацикливается. □

§ 1.3. Частично рекурсивные функции

§ 1.3.1. Примитивно рекурсивные функции

Напомним, что каждая функция характеризуется количеством аргументов — местностью. Следует отметить, что даже сходные функции разной местности следует считать различными. Например, функция $f^{(1)}$ одного аргумента, которая всегда принимает значение 0, отличается от функции $g^{(2)}$ двух аргументов, которая также всегда равна 0.

Определение 35 (Базисные рекурсивные функции)**.** *Базисными рекурсивными функциями называются:*

1) *функция без аргументов* 0*, значение которой равняется нулю;*
2) *функция с одним аргументом* $s(x)$*, значение которой равняется* $x+1$*;*
3) *всевозможные проектирующие* n*-местные функции* $\mathrm{id}_n^m(x_1, \ldots, x_n)$*,* $n, m \in \omega$*,* $0 < m \leqslant n$*, значение которых равняется* x_m*.*

Определение 36 (Суперпозиция)**.** *Пусть даны* m*-местная функция* g *и* n*-местные функции* $f_1, \ldots, f_m$*. Функция* h *получена из* $g, f_1, \ldots, f_m$ *суперпозицией (или подстановкой), если* $h(x_1, \ldots, x_n) = g(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$*. Суперпозиция обозначается так:* $h = g(f_1, \ldots, f_m)$*.*

Определение 37 (Примитивная рекурсия)**.** *Пусть даны* $n-1$*-местная функция* f *и* $n+1$*-местная функция* g*. Функция* h *получена из* f *и* g *примитивной рекурсией, если* $h(x_1, \ldots, x_{n-1}, 0) = f(x_1, \ldots, x_{n-1})$ *и* $h(x_1, \ldots, x_{n-1}, i+1) = g(x_1, \ldots, x_{n-1}, i, h(x_1, \ldots, x_{n-1}, i))$ *для каждого* $i \in \omega$*. Функция* h *обозначается в этом случае с помощью* $\mathrm{Pr}[f, g]$*.*

Определение 38 (Примитивно рекурсивная функция)**.** *Функция, которая может быть получена из базисных рекурсивных функций при помощи*

операторов суперпозиции и примитивной рекурсии называется п р и м и т и в н о р е к у р с и в н о й.

Определение 39 (Характеристическая функция)**.** *Пусть R — n-местное отношение.* Х а р а к т е р и с т и ч е с к о й ф у н к ц и е й *отношения R называется n-местная функция I_R такая, что $I_R(\bar{x}) = 1$, если $R(\bar{x})$ выполнено, и $I_R(\bar{x}) = 0$, если $R(\bar{x})$ не выполнено.*

Предложение 34. *Следующие функции являются примитивно рекурсивными:*

1) *нульместные константы:* $1^{(0)}$, $2^{(0)}$, ...;
2) *одноместные константы:* $0^{(1)}(x)$, $1^{(1)}(x)$, ...;
3) *характеристическая функция нуля* $\mathrm{not}(x)$, *превращающая нуль в единицу, а остальные числа — в нуль;*
4) *характеристическая функция множества положительных чисел:* $\mathrm{test}(x)$;
5) *сложение;*
6) *умножение;*
7) *возведение в степень;*
8) $\mathrm{dec}(x)$ — *ограниченный декремент:* $\mathrm{dec}(x+1) = x$, $\mathrm{dec}(0) = 0$;
9) *ограниченное вычитание:* $\lfloor x - y \rfloor$;
10) $\mathrm{less}(x, y)$ — *характеристическая функция отношения «меньше»;*
11) $\mathrm{leq}(x, y)$ — *характеристическая функция отношения «меньше или равно»;*
12) $\mathrm{eq}(x, y)$ — *характеристическая функция отношения «равно».*

Доказательство. 1) $1 = s(0)$, $2 = s(1)$ и т.д.

2) Рассмотрим одноместную функцию $f = \mathrm{Pr}[0, \mathrm{id}_2^2]$. Тогда

$$f(0) = 0 \text{ и } f(i+1) = \mathrm{id}_2^2(i, f(i)) = f(i) = 0.$$

Следовательно, $f = 0^{(1)}$. Аналогично для остальных чисел: $i^{(1)} = \mathrm{Pr}[i^{(0)}, \mathrm{id}_2^2]$.

3) $\text{not} = \Pr[1, 0(\text{id}_2^1)]$. Тогда

$$\text{not}(0) = 1, \quad \text{not}(x+1) = 0^{(1)}(\text{id}_2^1(x, \text{not}(x))) = 0.$$

4) $\text{test} = \text{not}(\text{not})$.

5) Сначала построим трехместную функцию $s^{(3)} = s(\text{id}_3^3)$, для которой $s^{(3)}(x, y, z) = z + 1$. Теперь рассмотрим $\text{sum} = \Pr[\text{id}_1^1, s^{(3)}]$. Покажем, что $\text{sum}(x, y) = x + y$ индукцией по y. В самом деле, по определению примитивной рекурсии получаем: $\text{sum}(x, 0) = \text{id}_1^1(x) = x = x + 0$. Предположим, что для y утверждение доказано. Тогда

$$\text{sum}(x, y+1) = s^{(3)}(x, y, \text{sum}(x, y)) = \\ = s^{(3)}(x, y, x+y) = (x+y) + 1 = x + (y+1).$$

6) Сначала построим $\text{sum}^{(3)} = \text{sum}(\text{id}_1^3, \text{id}_3^3)$ — трехместную функцию, результатом которой является сумма первого и третьего аргументов. Пусть теперь $\text{prod} = \Pr[0^{(1)}, \text{sum}^{(3)}]$. Покажем, что $\text{prod}(x, y) = x \times y$ индукцией по y. По определению примитивной рекурсии получаем: $\text{prod}(x, 0) = 0^{(1)}(x) = 0 = x \times 0$. Предположим, что для y утверждение доказано. Тогда

$$\text{prod}(x, y+1) = \text{sum}^{(3)}(x, y, \text{prod}(x, y)) = \\ = \text{sum}^{(3)}(x, y, x \times y) = x \times y + x = x \times (y+1).$$

7) Аналогично предыдущим:

$$\text{prod}^{(3)} = \text{prod}(\text{id}_1^3, \text{id}_3^3), \quad \text{power} = \Pr[1^{(1)}, \text{prod}^{(3)}].$$

8) Пусть $\text{dec} = \Pr[0^{(0)}, \text{id}_2^1]$. Тогда

$$\text{dec}(0) = 0, \quad \text{dec}(x+1) = \text{id}_2^1(x, \text{dec}(x)) = x.$$

9) Аналогично пунктам про сложение и умножение легко можно показать, что $\lfloor x - y \rfloor = \Pr[\text{id}_1^1, \text{dec}(\text{id}_3^3)]$.

10) Положим $\text{less} = \text{test}(\lfloor \text{id}_2^2 - \text{id}_2^1 \rfloor)$. Тогда $\text{less}(x, y)$ будет равно единице, если разность $y - x$ положительна, то есть $x < y$, или нулю иначе.

11) $\text{leq} = \text{not}(\text{less}(\text{id}_2^2, \text{id}_2^1))$.

12) $\text{eq} = \text{leq}(\text{id}_2^1, \text{id}_2^2) \times \text{leq}(\text{id}_2^2, \text{id}_2^1)$. □

Предложение 35. *Следующая функция* if *является примитивно рекурсивной:*

$$\text{if}(x, y, z) = \begin{cases} y, & \text{если } x \neq 0; \\ z, & \text{в противном случае.} \end{cases}$$

Доказательство. Нетрудно заметить, что

$$\text{if} = \text{test}(\text{id}_3^1) \times \text{id}_3^2 + \text{not}(\text{id}_3^1) \times \text{id}_3^3 . \qquad \square$$

Определение 40 (Ограниченная минимизация)**.** *Пусть $f(x_1, \ldots, x_n)$ — n-местная функция. n-местная функция h получена из f с помощью ограниченной минимизации: $h = \mu_{\lim} f$, если $h(x_1, \ldots, x_{n-1}, v) = y$, когда y — наименьшее число меньшее v, для которого $f(x_1, \ldots, x_{n-1}, y) = 0$, или же $h(x_1, \ldots, x_{n-1}, v) = v$, если такого y не существует.*

Предложение 36. *Если функция f является примитивно рекурсивной, то функция $h = \mu_{\lim} f$ тоже является примитивно рекурсивной.*

Доказательство. Определим две функции $f_1(\bar{x}, v) = v$ и

$$f_2(\bar{x}, v, u, w) = \begin{cases} u, & \text{если } f(\bar{x}, u) = 0 \text{ и } w = v; \\ w, & \text{в противном случае.} \end{cases}$$

Заметим, что условие имеет примитивно рекурсивную характеристическую функцию:

$$\text{not}(f(\bar{x}, u)) \times \text{eq}(w, v),$$

следовательно по предложению 35 на предыдущей странице функция f_2 тоже является примитивно рекурсивной. Пусть $g = \text{Pr}[f_1, f_2]$. Покажем, что $h(\bar{x}, v) = g(\bar{x}, v, v)$.

Предположим, что $h(\bar{x}, v) = v$. Это означает, что $f(\bar{x}, u) \neq 0$ для всех $u < v$. По определению примитивной рекурсии получаем

$$g(\bar{x}, v, 0) = f_1(\bar{x}, v) = v;$$

$$g(\bar{x}, v, u+1) = f_2(\bar{x}, v, u, g(\bar{x}, v, u)) = g(\bar{x}, v, u) = v.$$

Последнее верно, поскольку $f(\bar{x}, u) \neq 0$, следовательно, $f_2(\bar{x}, v, u, w) = w$.

Пусть теперь $h(\bar{x}, v) = v' < v$. Следовательно, v' — наименьшее натуральное число меньшее v, для которого $f(\bar{x}, v') = 0$. Для $u \leqslant v'$ как и в предыдущем случае получаем $g(\bar{x}, v, u) = v$. При $u = v'$ будет выполнено

$$g(\bar{x}, v, v'+1) = f_2(\bar{x}, v, v', g'(\bar{x}, v, v')) = f_2(\bar{x}, v, v', v) = v'.$$

При $u > v'$ по индукции получим $g(\bar{x}, v, u+1) = v'$:

$$g(\bar{x}, v, u+1) = f_2(\bar{x}, v, u, g(\bar{x}, v, u)) = f_2(\bar{x}, v, u, v') = v'.$$

Последнее выполнено, поскольку $v' \neq v$. $\square$

Определение 41 (Минимизация на интервале)**.** *Пусть $f(x_1, \ldots, x_n)$ — n-местная функция. $n+1$-местная функция h получена из f минимизацией на интервале: $h = \mu_{\text{int}} f$, если $h(x_1, \ldots, x_{n-1}, a, b) = y$, когда y — наименьшее число из интервала $[a, b)$, для которого $f(x_1, \ldots, x_{n-1}, y) = 0$. Если такого y не существует, то $h(x_1, \ldots, x_{n-1}, a, b) = b$. При $b < a$ значение h может быть произвольным.*

Следствие 37. *Если функция f является примитивно рекурсивной, то функция $h = \mu_{\text{int}} f$ тоже является примитивно рекурсивной.*

Доказательство. Очевидно, что функция $g(\bar{x}, z, y) = f(\bar{x}, y + z)$ является примитивно рекурсивной. Пусть $h' = \mu_{\lim} g$. Тогда $h(\bar{x}, a, b) = h'(\bar{x}, a, \lfloor b - a \rfloor) + a$.

В самом деле, пусть функция f не принимает нулевого значения на $[a, b)$. Тогда по определению ограниченной минимизации $h'(\bar{x}, a, \lfloor b - a \rfloor) = \lfloor b - a \rfloor$, так как $g(\bar{x}, a, u) = f(\bar{x}, u + a) \neq 0$ при $u < \lfloor b - a \rfloor$. Следовательно, $h(\bar{x}, a, b) = h'(\bar{x}, a, \lfloor b - a \rfloor) + a = \lfloor b - a \rfloor + a = b$.

Если y — наименьшее значение интервала $[a, b)$, для которого $f(\bar{x}, y) = 0$, то $y - a$ — наименьшее число, которое меньше $\lfloor b - a \rfloor$ и для которого $g(\bar{x}, a, y) = 0$. Следовательно, $h'(\bar{x}, a, \lfloor b - a \rfloor) = y - a$, поэтому $h(\bar{x}, a, b) = y$. □

Предложение 38. *Пусть характеристическая функция $I_R(\bar{x}, y)$ отношения $R(\bar{x}, y)$ является примитивно рекурсивной. Тогда функция $I_P(\bar{x}, a, b)$ отношения*

$$P(\bar{x}, a, b) = \{(\bar{x}, a, b) : \text{существует } y \in [a, b), \text{ для которого выполнено } R(\bar{x}, y)\}$$

тоже является примитивно рекурсивной.

Доказательство. Рассмотрим следующую примитивно рекурсивную функцию:

$$h(\bar{x}, a, b) = \text{less}(a, b) \times \text{less}(\mu_{z < \lfloor b - a \rfloor} \text{not}(f(\bar{x}, z + a)) + a, b).$$

Покажем, что она и является искомой.

Предположим, выполнено $P(\bar{x}, a, b)$. По определению это означает, что существует $y \in [a, b)$, для которого выполнено $R(\bar{x}, y)$. Из этого в частности следует, что $a < b$ (иначе интервал будет пустым и такого y существовать не может), поэтому $\text{less}(a, b) = 1$. Поскольку $f(\bar{x}, y) = 0$ и $a \leqslant y < b$, то при $z =$

$y-a < \lfloor b-a \rfloor$ мы получим $f(\bar{x}, z+a) = 0$. Поэтому результат минимизации будет меньше $\lfloor b-a \rfloor$, следовательно, значение второй функции less тоже будет равно 1 и $h(\bar{x}, a, b) = 1$.

Пусть теперь $P(\bar{x}, a, b)$ не выполнено. Если $a \geqslant b$, то $\mathrm{less}(a, b) = 0$ и произведение тоже равно 0. Если $a < b$, то ни одно из чисел y интервала $[a, b)$ не удовлетворяет условию $R(\bar{x}, y)$. Следовательно, для каждого натурального z меньшего $\lfloor b-a \rfloor$ получаем $f(\bar{x}, z+a) = 0$. Поэтому результат ограниченной минимизации будет равен $\lfloor b-a \rfloor$ и оба аргумента второй функции less будут равны. Таким образом, снова получаем $h(\bar{x}, a, b) = 0$. □

Определение 42. *Будем обозначать функцию, построенную в доказательстве предложения 38 на противоположной странице с помощью* $\mathrm{exists}_{y\in[a,b)}\, f(\bar{x}, y)$.

Предложение 39. *Следующие функции являются примитивно рекурсивными:*

1) $\left[\frac{x}{y}\right]$ — *целочисленное деление;*
2) *остаток от деления;*
3) *характеристическая функция делимости;*
4) $\mathrm{prime}(x)$ — *характеристическая функция множества простых чисел;*
5) $\mathrm{pprime}(x, p)$ — *характеристическая функция отношения «x — степень простого числа p»;*
6) $\langle x, y\rangle$ — *канторова нумерующая функция;*
7) $\langle z\rangle_1$, $\langle z\rangle_2$ — *обращения канторовой нумерующей функции.*

Доказательство. 1) Чтобы функция была всюду определенной будем полагать, что результатом деления на нуль будет нуль. Пусть $f^{(3)} = \mu_{\lim}\,\mathrm{leq}(\mathrm{id}_3^2 \times(\mathrm{id}_3^3+1), \mathrm{id}_3^1)$. Тогда $f(x, y, x+1)$ (при $y \neq 0$) — это наименьшее $u \leqslant x$, для которого не выполнено $y(u+1) \leqslant x$, то есть $y(u+1) > x$. Другими словами $yu \leqslant x < y(u+1)$, что и означает $u = \left[\frac{x}{y}\right]$. Следовательно,

$$\left[\frac{x}{y}\right] = \mathrm{test}(y) \times f(x, y, x+1).$$

2) Аналогично считаем, что $x \bmod 0 = 0$. Тогда

$$x \bmod y = \operatorname{test}(y) \times \left| x - \left[\frac{x}{y}\right] \times y \right|.$$

3) Очевидно, $(x \mid y) = \operatorname{not}(x \bmod y)$.

4) Пусть $f^{(3)} = \mu_{\text{int}}(\mathrm{id}_3^1 \bmod \mathrm{id}_3^2)$. Покажем, что $\operatorname{prime}(x) = \operatorname{leq}(2, x) \times \operatorname{eq}(f(x, 2, x), x)$. Если x — простое, то $x \geqslant 2$ и у него нет делителей на интервале $[2, x)$. Следовательно, $\operatorname{leq}(2, x) = 1$ и $f(x, 2, x) = x$, что и означает $\operatorname{prime}(x) = 1$.

В противном случае при $x < 2$ сразу получаем $\operatorname{leq}(2, x) = 0$ и $\operatorname{prime}(x) = 0$. Если же $x \geqslant 2$, то у него существует наименьший делитель y на интервале $[2, x)$, поэтому $f(x, 2, x) = y \neq x$ и $\operatorname{prime}(x) = 0$.

5) Число x является степенью простого числа p, если и только если $x > 0$ и не существует делителя x большего 1, который не делился бы на p. Построим функцию:

$$\operatorname{pprime}(x) = \operatorname{test}(x) \times \operatorname{prime}(p) \times \\ \times \operatorname{not}\operatorname{exists}_{y \in [2,x)}(\operatorname{not}(x \bmod y) \times \operatorname{test}(y \bmod p)).$$

6) $\langle x, y\rangle = \left[\dfrac{(x+y)(x+y+1)}{2}\right] + x$, нужно только заметить, что числитель дроби в скобках всегда будет четным числом, поэтому результат целочисленного деления равен результату «настоящего» деления.

7) Сначала определим вспомогательную функцию

$$f(z) = \mu_{u<z+1} \operatorname{leq}((u+1) \times (u+2), z).$$

Очевидно, что $f(z)$ — наибольшее u, для которого $u(u+1) \leqslant z$. Тогда $\langle z\rangle_1 = z - f(z) \times (f(z)+1)$, $\langle z\rangle_2 = f(z) - \langle z\rangle_1$. □

Предложение 40. *Для n-ого простого числа p_n выполнено $p_n \leqslant 2^{2^n}$.*

Доказательство. Обозначим

$$z_n = \prod_{i<n} 2^{2^i}.$$

Индукцией по n докажем, что $z_n + 1 \leqslant 2^{2^n}$. Для $n = 0$ получаем $z_0 + 1 = 1 + 1 = 2 = 2^{2^0}$. Если для n неравенство $z_n + 1 \leqslant 2^{2^n}$ доказано, то

$$z_{n+1} + 1 = 1 + \prod_{i<n+1} 2^{2^i} = 1 + 2^{2^n} \prod_{i<n} 2^{2^i} \leqslant \\ \leqslant 1 + 2^{2^n} z_n \leqslant 1 + 2^{2^n}(2^{2^n} - 1) = 2^{2^{n+1}} - 2^{2^n} + 1 \leqslant 2^{2^{n+1}}.$$

Теперь индукцией по n докажем утверждение теоремы. Для $n = 0$ утверждение очевидно:

$$p_0 = 2^{2^0} = 2.$$

Предположим, для n утверждение доказано. Заметим, что число $x = 1 + \prod\limits_{i<n+1} p_i$ не может делиться ни на одно из чисел $p_0, \ldots, p_n$, поэтому при разложении x на простые множители мы обязательно получим простое число, которое будет больше p_n, но не больше x. Следовательно, $p_{n+1} \leqslant x$. Далее получаем:

$$p_{n+1} \leqslant x = 1 + \prod_{i<n+1} p_i \leqslant 1 + \prod_{i<n+1} 2^{2^i} = 1 + z_{n+1} \leqslant 2^{2^{n+1}}. \qquad \square$$

Предложение 41. *Следующие функции являются примитивно рекурсивными:*

1) $\mathrm{nextprime}(x)$ — *наименьшее простое число, большее* x*;*
2) $\mathrm{iprime}(x)$ — x*-ое по счету простое число: нулевое* — 2, *первое* — 3 *и т.д.;*
3) $\mathrm{ind}(x, p)$ — *степень простого числа* p *в разложении* x *на простые множители.*

Доказательство. 1) Из предложения 40 на предыдущей странице получаем, следующее за x простое число не превосходит 2^{2^x}, поэтому

$$\mathrm{nextprime}(x) = \mathrm{leq}(2, x) \times (\mu_{y<2^{2^x}} \,\mathrm{not}\,\mathrm{prime}(x + 1 + y) + x + 1).$$

2) $\mathrm{iprime}(x) = \mathrm{Pr}[2, \mathrm{nextprime}(\mathrm{id}_2^2(i, u))]$.

3) $\mathrm{ind}(x, p) = \mathrm{test}(x) \times \mathrm{prime}(p) \times \mu_{u<x}(\mathrm{not}(\dfrac{x}{p^u} \bmod p))$. $\square$

Определение 43 (Совместная рекурсия)**.** *Пусть даны* $n-1$*-местные функции* $f_1, \ldots, f_k$ *и* $n+k$*-местные функции* $g_1, \ldots, g_k$*. Функции* $h_1, \ldots, h_k$ *получены из* $f_1, \ldots, f_k, g_1, \ldots, g_k$ *совместной рекурсией, если*

$$h_j(x_1, \ldots, x_{n-1}, 0) = f_j(x_1, \ldots, x_{n-1})$$

для $j = 1, \ldots, k$ *и*

$$h_j(x_1, \ldots, x_{n-1}, i+1) = \\ = g_j(x_1, \ldots, x_{n-1}, i, h_1(x_1, \ldots, x_{n-1}, i), \ldots, h_k(x_1, \ldots, x_{n-1}, i))$$

для $j = 1, \ldots, k$ для каждого $i \in \omega$. Функция h_j обозначается с помощью $\mathrm{Pr}_j[f_1, \ldots, f_k, g_1, \ldots, g_k]$.

Предложение 42. *Если функции $f_1, \ldots, f_k$ и $g_1, \ldots, g_k$ являются примитивно рекурсивными, то и функции $h_j = \mathrm{Pr}_j[f_1, \ldots, f_k, g_1, \ldots, g_k]$ будут примитивно рекурсивными.*

Доказательство. Определим две функции: $n-1$-местную функцию $f(\bar{x})$

$$f(\bar{x}) = \langle f_1(\bar{x}), \ldots, f_k(\bar{x}) \rangle,$$

и $n+1$-местную функцию $g(\bar{x}, i, v)$

$$g(\bar{x}, i, v) = \left\langle g_1(\bar{x}, i, \langle v \rangle_k^1, \ldots, \langle v \rangle_k^k), \ldots, g_k(\bar{x}, i, \langle v \rangle_k^1, \ldots, \langle v \rangle_k^k) \right\rangle$$

Построим $h = \mathrm{Pr}[f, g]$. Индукцией по i покажем, что $h_j(\bar{x}, i) = \langle h(\bar{x}, i) \rangle_k^j$. Для $i = 0$ и то и другое значение равны $f_j(\bar{x})$, поэтому утверждение выполняется.

Предположим, что для i утверждение доказано. По построению

$$\langle h(\bar{x}, i+1) \rangle_k^j = g_j(\bar{x}, i, \langle v \rangle_k^1, \ldots, \langle v \rangle_k^k),$$

где $v = h(\bar{x}, i)$. По индукционному предположению

$$\langle v \rangle_k^j = \langle h(\bar{x}, i) \rangle_k^j = h_j(\bar{x}, i).$$

Поэтому

$$g_j(\bar{x}, i, \langle v \rangle_k^1, \ldots, \langle v \rangle_k^k) = g_j(\bar{x}, i, h_1(\bar{x}, i), \ldots, h_k(\bar{x}, i)) = h_j(\bar{x}, i+1),$$

что и требовалось. □

§ 1.3.2. Функция Аккермана

Легко заметить, что для любой (сколь угодно быстро растущей) примитивно рекурсивной функции f, можно найти примитивно рекурсивную функцию, которая растет еще быстрее. В качестве такой можно, скажем, рассмотреть $2^{f(x)}$. Но оказывается, что для этого роста существует некоторая верхняя грань.

Определение 44 (Функция Аккермана)**.** *Функцией Аккермана называется следующая двухместная функция:*

$$A(x,y) = \begin{cases} y+1, & \text{при } x = 0; \\ A(x-1,1), & \text{при } x > 0 \text{ и } y = 0; \\ A(x-1, A(x,y-1)), & \text{при } x > 0 \text{ и } y > 0. \end{cases}$$

Пример 19. *Из определения непосредственно получаем* $A(0,y) = y+1$.

Для $x = 1$ *получаем* $A(1,0) = A(0,1) = 2$, $A(1,y+1) = A(0,A(1,y)) = A(1,y)+1$. *Следовательно,* $A(1,y) = 2+y$.

Для $x = 2$ *получаем* $A(2,0) = A(1,1) = 3$, $A(2,y+1) = A(1,A(2,y)) = A(2,y)+2$. *Следовательно,* $A(2,y) = 3+2y$.

Для $x = 3$ *получаем* $A(3,0) = A(2,1) = 5$, $A(3,y+1) = A(2,A(3,y)) = 3+2A(3,y)$. *Нетрудно проверить, что* $A(3,y) = 2^{y+3}-3$. *Для* $y = 0$ *это тривиально. Если для* y *это уже доказано, то для* $y+1$ *имеем*

$$A(3,y+1) = 3+2A(3,y) = 3+2(2^{y+3}-3) = \\ = 3+2\times 2^{y+3} - 2\times 3 = 2^{y+1+3} - 3.$$

Из примера нетрудно заметить, что с увеличением первого аргумента функция Аккермана растет чрезвычайно быстро.

Лемма 43. $A(x,y) > y$.

Доказательство. Для $x = 0$ из определения сразу получаем $A(0,y) = y+1 > y$.

Для $x+1$ используем индукцию по y:

$$A(x+1,0) = A(x,1) > 1 > 0;$$

$$A(x+1,y+1) = A(x,A(x+1,y)) > A(x+1,y) > y.$$

Неравенство $A(x,z) > z$ выполнено по индукционному предположению (для $z = 1$ и $z = A(x+1,y)$). □

Лемма 44. $A(x,y) < A(x,y+1)$.

Доказательство. Для $x = 0$ из определения сразу получаем $A(0,y) = y+1 < y+2 = A(x,y+1)$.

Для $x+1$, пользуясь неравенством из леммы 43, получаем

$$A(x+1,y) < A(x,A(x+1,y)) = A(x+1,y+1),$$

□

Следствие 45. *Если* $y_1 < y_2$, *то* $A(x, y_1) < A(x, y_2)$.

Лемма 46. $A(x, y) < A(x + 1, y)$.

Доказательство. Для $x = 0$ из примера 19 на предыдущей странице получаем $A(0, y) = y + 1 < y + 2 = A(1, y)$.

Для $x + 1$, применим индукцию по y. При $y = 0$:

$$A(x + 1, 0) = A(x, 1) < A(x + 1, 1) = A(x + 2, 0),$$

здесь среднее неравенство является индукционным предположением.

Для $y + 1$ из индукционного предположения (по y) получаем:

$$A(x + 1, y) < A(x + 2, y).$$

Учитывая монотонность функции A по второму аргументу (следствие 45):

$$A(x, A(x + 1, y)) < A(x, A(x + 2, y)).$$

Теперь применим индукционное предположение по x:

$$A(x, A(x + 2, y)) < A(x + 1, A(x + 2, y)).$$

Используя выведенные неравенства, получим:

$$A(x + 1, y + 1) = A(x, A(x + 1, y)) < A(x, A(x + 2, y)) < \\ < A(x + 1, A(x + 2, y)) = A(x + 2, y + 1),$$

что и требуется. □

Следствие 47. *Если* $x_1 < x_2$, *то* $A(x_1, y) < A(x_2, y)$.

Лемма 48. $A(x, y + 1) \leqslant A(x + 1, y)$.

Доказательство. Для $x = 0$ из примера 19 на предыдущей странице сразу получаем $A(0, y + 1) = (y + 1) + 1 = y + 2 = A(1, y)$.

Для $x + 1$ применим индукцию по y. По определению:

$$A(x + 1, 1) = A(x + 2, 0).$$

Для $y + 1$ получаем такое индукционное предположение:

$$A(x + 1, y + 1) \leqslant A(x + 2, y),$$

а, используя монотонность функции A по второму аргументу (следствие 45), также будет

$$A(x, A(x + 1, y + 1)) \leqslant A(x, A(x + 2, y)).$$

Тогда получаем

$$A(x+1,y+2) = A(x,A(x+1,y+1)) \leqslant A(x,A(x+2,y)) < \\ < A(x+1,A(x+2,y)) = A(x+2,y+1),$$

где последнее неравенство получено из леммы 46 на предыдущей странице. □

Теорема 49. *Для каждой примитивно рекурсивной функции f существует натуральное число $n(f)$ такое, что $f(\bar{x}) \leqslant A(n(f), \max \bar{x})$. Здесь с помощью $\max \bar{x}$ обозначен наибольший элемент набора $\bar{x}$.*

Доказательство. Используем индукцию по построению функции f.

Пусть f является одной из базисных функций, тогда положим $n(f) = 0$. Для нулевой функции неравенство $0 \leqslant A(0, \max \bar{x})$ очевидно. Также тривиально получается

$$\mathrm{id}_n^m(\bar{x}) = x_m \leqslant \max \bar{x} < A(0, \max \bar{x}),$$

последнее — по лемме 43 на с. 53. Для функции s получаем

$$s(x) = x + 1 = A(0, x)$$

по определению функции Аккермана.

Допустим, функция f получена из $g, h_1, \ldots, h_m$ при помощи суперпозиции и для функций $g, h_1, \ldots, h_m$ утверждение уже доказано, то есть $n(g)$, $n(h_1), \ldots, n(h_m)$ уже найдены. Положим

$$n(f) = \max\{n(h_1) + 1, \ldots, n(h_m) + 1, n(g) + 2\}.$$

Тогда получаем

$$h_i(\bar{x}) \leqslant A(n(h_i), \max \bar{x}) \leqslant A(n(f) - 1, \max \bar{x})$$

для $i = 1, \ldots, m$. Далее

$$f(\bar{x}) = g(h_1(\bar{x}), \ldots, h_m(\bar{x})) \leqslant A(n(g), \max\{h_1(\bar{x}), \ldots, h_m(\bar{x})\}) \leqslant \\ \leqslant A(n(g), A(n(f) - 1, \max \bar{x})) \leqslant A(n(f) - 2, A(n(f) - 1, \max \bar{x})) = \\ = A(n(f) - 1, (\max \bar{x}) + 1) \leqslant A(n(f), \max \bar{x}).$$

Последнее неравенство получено из леммы 48 на предыдущей странице.

Пусть теперь функция f получена из h и g с помощью примитивной рекурсии и $n(h)$ и $n(g)$ уже найдены.

Сначала возьмем

$$n' = \max\{n(h), n(g) + 1\}$$

и покажем, что

$$f(\bar{x}, i) \leqslant A(n', \max \bar{x} + i).$$

Используем индукцию по i, базис:

$$f(\bar{x}, 0) = h(\bar{x}) = A(n(h), \max \bar{x}) \leqslant A(n', \max \bar{x} + 0).$$

Для $i + 1$ получаем

$$\begin{aligned} f(\bar{x}, i+1) = g(\bar{x}, i, f(\bar{x}, i)) &\leqslant A(n(g), \max(\bar{x}, i, f(\bar{x}, i))) \leqslant \\ &\leqslant A(n(g), \max(\bar{x}, i, A(n', \max \bar{x} + i))) \leqslant \\ &\leqslant A(n' - 1, A(n', \max \bar{x} + i)) = A(n', \max \bar{x} + i + 1). \end{aligned}$$

Пусть $n(f) = n' + 2$. При $i \leqslant \max \bar{x}$ будет выполнено $\max \bar{x} + i \leqslant 2 \max \bar{x}$ и тогда

$$\begin{aligned} f(\bar{x}, i) &\leqslant A(n', \max \bar{x} + i) \leqslant A(n', 2 \max \bar{x}) \leqslant \\ &\leqslant A(n', A(2, \max \bar{x})) \leqslant A(n', A(n' + 1, \max \bar{x})) \leqslant A(n' + 1, \max \bar{x} + 1) \leqslant \\ &\leqslant A(n' + 2, \max \bar{x}) = A(n(f), \max \bar{x}) = A(n(f), \max(\bar{x}, i)). \end{aligned}$$

Здесь мы воспользовались тем, что $n' \geqslant 1$, поэтому $n' + 1 \geqslant 2$ и монотонностью функции A.

При $i > \max \bar{x}$ будет $\max \bar{x} + i \leqslant 2i$ и тогда по аналогии получается

$$f(\bar{x}, i) \leqslant A(n(f), i) = A(n(f), \max(\bar{x}, i)).$$

□

Между тем, легко понять, что функция Аккермана вычислима, алгоритм непосредственно следует из определения.

Теорема 50. *Существует счетчиковая машина, вычисляющая функцию Аккермана.*

Доказательство. Будем использовать машину с шестью счетчиками (рис. 1.1 на противоположной странице). Шестой будем применять как вспомогательный для тех программ, которые применяются при построении.

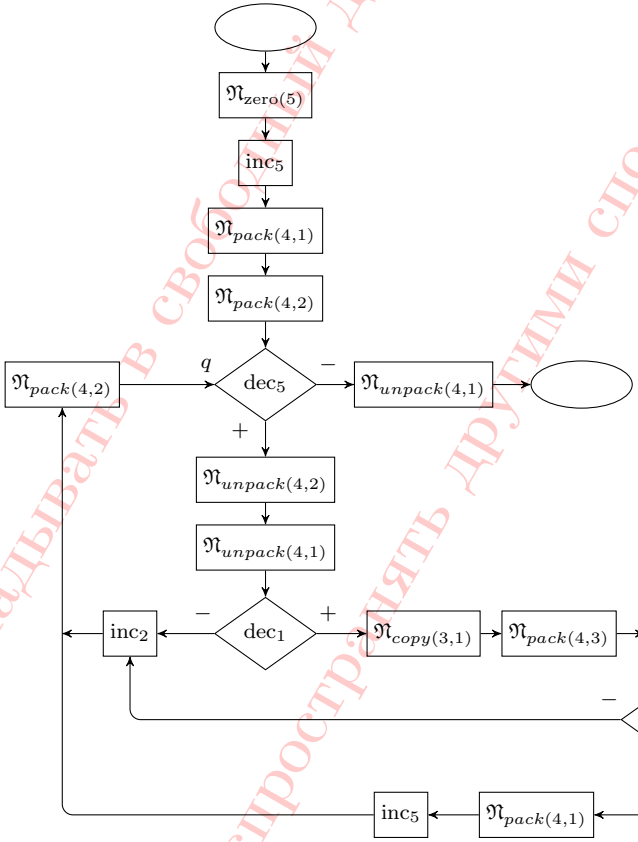


Рис. 1.1: Вычисление функции Аккермана.

Докажем, что данная счетчиковая машина действительно вычисляет функцию Аккермана. Для этого покажем, что если в счетчике 4 в состоянии q хранится код некоего набора

$$\langle x_1, x_2, \ldots, x_{k-2}, x_{k-1}, x_k, w\rangle \tag{1.6}$$

то результат будет равен

$$A(x_1, A(x_2, A(\ldots, A(x_{k-1}, A(x_k, w))\ldots))). \tag{1.7}$$

Для этого в программе (согласно определению 44 на с. 53) вместо кода

$$\langle x_1, x_2, \ldots, x_{k-2}, x_{k-1}, x_k, w\rangle$$

кладется в счетчик 4 один из трех кодов:

$$\langle x_1, x_2, \ldots, x_{k-2}, x_{k-1}, w+1\rangle$$

при $x_k = 0$,

$$\langle x_1, x_2, \ldots, x_{k-2}, x_{k-1}, x_k - 1, 1\rangle$$

при $x_k > 0$ и $w = 0$,

$$\langle x_1, x_2, \ldots, x_{k-2}, x_{k-1}, x_k - 1, x_k, w - 1\rangle$$

в остальных случаях. После этого повторяется та же процедура. Счетчик 5 содержит количество функций A, которые надо вычислить в (1.7), остальные счетчики используются как рабочие для выполнения действий.

Из проводимых преобразований и определения 44 на с. 53 видно, что значение выражения (1.7) не меняется. Поскольку $A(u, v) \geqslant v+1$, а в начале набор (1.6) имеет вид $\langle x, y\rangle$, то количество функций A в (1.7) не должно быть больше $A(x, w)$, а длина набора (1.6) не превосходит $A(x, w) + 1$.

Рассмотрим наборы (1.6) без последнего элемента:

$$\langle x_1, x_2, \ldots, x_{k-2}, x_{k-1}, x_k\rangle\,. \tag{1.8}$$

Их длины ограничены значением $A(x, w)$, элементы этих наборов не превосходят x (добавляемые новые элементы не превосходят уже имеющиеся), поэтому общее их количество конечно. Каждый следующий набор (1.8) как видно из преобразований лексикографически меньше предыдущего. Следовательно, рано или поздно алгоритм остановится.

Поскольку это произойдет, когда счетчик 5 станет равен 0, то выражение (1.7) не будет содержать ни одной функции A, а поскольку значение этого выражения всегда остается постоянным, то набор (1.6) будет состоять из единственного элемента w равного $A(x, y)$. □

Предложение 51. *Функция Аккермана не является примитивно рекурсивной.*

Доказательство. Предположим противное: является. Тогда примитивно рекурсивной будет и функция $f = A(\mathrm{id}_1^1, \mathrm{id}_1^1)$. По теореме 49 на с. 55 получаем, что для некоторого n должно выполняться $f(x) \leqslant A(n, x)$ для всех x. При $x = n + 1$ мы бы получили

$$A(n+1, n+1) = f(n+1) \leqslant A(n, n+1),$$

что противоречит лемме 46 на с. 54. □

Предложение 52. *Функции $A_x(y) = A(x, y)$, полученные из функции Аккермана фиксированием первого аргумента являются примитивно рекурсивными.*

Доказательство. Применим индукцию по x. Для функции $A_0(y) = A(0, y) = y + 1$ это очевидно: $A_0 = s$.

Пусть функция $A_x(y)$ является примитивно рекурсивной. $A_x(1)$ — это некоторая константа. Покажем, что $A_{x+1} = \mathrm{Pr}[A_x(1), A_x(\mathrm{id}_2^2)]$:

$$A_{x+1}(0) = A(x+1, 0) = A(x, 1) = A_x(1);$$

$$A_{x+1}(y+1) = A(x+1, y+1) = A(x, A(x+1, y)) = \\ = A_x(A_{x+1}(y)) = A_x(\mathrm{id}_2^2(y, A_{x+1}(y))).$$

□

§ 1.3.3. Минимизация, частично и общерекурсивные функции функции

Определение 45 (Минимизация)**.** *Пусть $f(x_1, \ldots, x_{n-1}, x_n)$ — n-местная функция. Частичная $n-1$-местная функция h получена из f с помощью (полной) минимизации: $h = \mu f$, если $h(x_1, \ldots, x_{n-1}) = y$, когда y — наименьшее число, для которого $f(\bar{x}, y) = 0$. Если такого y не существует, то $h(x_1, \ldots, x_n)$ неопределено.*

Определение 46 (Частично рекурсивная, общерекурсивная функция)**.** *(Частичная) функция, которая может быть получена из базисных рекурсивных функций при помощи операторов суперпозиции, примитивной рекурсии и минимизации называется частично рекурсивной функцией. Если частично рекурсивная функция является*

всюду определенной, то она называется о б щ е р е к у р с и в н о й.

Следствие 53. *Каждая примитивно рекурсивная функция является общерекурсивной.*

Обратное неверно: покажем, что функция Аккермана является общерекурсивной.

Предложение 54. *Функция Аккермана является общерекурсивной.*

Доказательство. Покажем, что преобразование набора

$$\langle x_1, x_2, \ldots, x_{k-2}, x_{k-1}, x_k, w\rangle$$

из доказательства теоремы 50 на с. 56 может быть выполнено с помощью примитивно рекурсивной функции g_1. Пусть v — код этого набора:

$$g_1(x,y,i,v,z) = \begin{cases} \left\langle \langle v\rangle_3^1, \langle v\rangle_3^3 + 1\right\rangle, & \text{если } \langle v\rangle_3^2 = 0, \\ \left\langle \langle v\rangle_3^1, \langle v\rangle_3^2 - 1, 1\right\rangle, & \text{если } \langle v\rangle_3^2 > 0 \text{ и } \langle v\rangle_3^3 = 0, \\ \left\langle \langle v\rangle_3^1, \langle v\rangle_3^2 - 1, \langle v\rangle_3^2, \langle v\rangle_3^1 - 1\right\rangle, & \text{иначе.} \end{cases}$$

Начальное значение этого набора также можно вычислить с помощью примитивно рекурсивной функции $f_1(x,y) = \langle 0, x, y\rangle$.

Кроме самого набора меняется и его длина. Она тоже вычисляется с помощью пары функций g_2 (для следующего момента) и f_2 (для начального):

$$g_2(x,y,i,v,z) = \begin{cases} z-1, & \text{если } \langle v\rangle_3^2 = 0, \\ z, & \text{если } \langle v\rangle_3^2 > 0 \text{ и } \langle v\rangle_3^3 = 0, \\ z+1, & \text{иначе;} \end{cases}$$

$$f_2(x,y) = 1.$$

Построим функции h_1 и h_2 с помощью совместной рекурсии:

$$h_1(x,y,i) = \mathrm{Pr}_1[f_1, f_2, g_1, g_2];$$

$$h_2(x,y,i) = \mathrm{Pr}_2[f_1, f_2, g_1, g_2].$$

Если длина набора стала равна 0, то его единственный элемент и является значением функции Аккермана. Поэтому

$$A(x,y) = \langle h_1(x,y,\mu_i h_2(x,y,i))\rangle_1.$$

□

Поскольку в результате минимизации могут получиться частичные функции, то нужно уточнить, как такие функции используются при применении операторов суперпозиции, примитивной рекурсии и минимизации.

Определение 47. *Пусть $g^{(m)}$, $f_1, \ldots, f_m$ — частичные функции. Частичная функция h, полученная из $g, f_1, \ldots, f_m$ суперпозицией, определена на $\bar{y}$ и имеет значение u, если $f_i(\bar{y})$ определено и равно z_i для $i = 1, \ldots, m$, $g(z_1, \ldots, z_m)$ определено и равно u.*

Определение 48. *Пусть $f^{(n-1)}$, $g^{(n+1)}$ — частичные функции. Частичная функция h, полученная из f и g примитивной рекурсией, определена на $(\bar{y}, i)$ и имеет значение u, если выполнено одно из двух условий:*

1) *$i = 0$, $f(\bar{y})$ определено и равно u;*
2) *$i = j + 1$, $h(\bar{y}, j)$ определено по индукции и равно v, $g(\bar{y}, j, v)$ определено и равно u.*

Определение 49. *Пусть f — частичная функция. Частичная функция h, полученная из f минимизацией определена на $(\bar{x})$ и имеет значение u, если значения $f(\bar{x}, 0)$, …, $f(\bar{x}, u-1)$ определены и не равны нулю, а $f(\bar{x}, u)$ определено и равно нулю.*

§ 1.3.4. Вычислимость частично рекурсивных функций

Теорема 55. *Каждая частично рекурсивная функция вычислима на счетчиковой машине.*

Доказательство. Индукция по построению. Базисные рекурсивные функции тривиально вычислимы.

Для вычисления 0 достаточно обнулить первый счетчик:

$$q_0 \to \mathrm{dec}_1, q_0 : q_f.$$

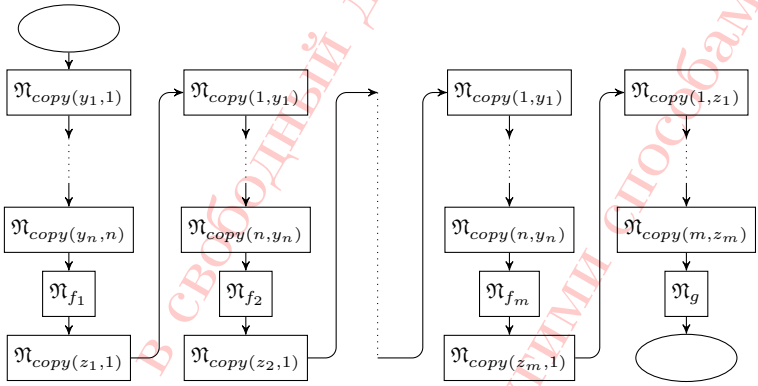
Для вычисления $s(x)$ нужно инкрементировать первый счетчик:

$$q_0 \to \mathrm{inc}_1, q_f.$$

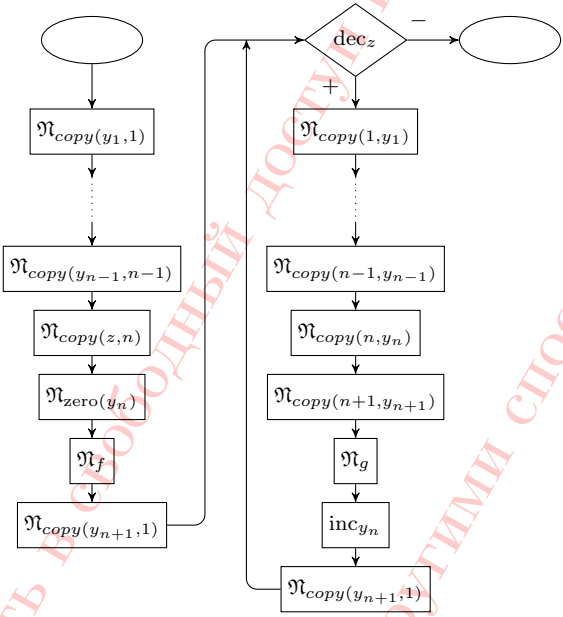
При вычислении функции id_n^m требуется просто переместить содержимое m-го счетчика в первый (машина $\mathfrak{N}_{\mathrm{move}(1,m)}$ из предложения 10 на с. 12).

Если вычислимы функции $g^{(m)}, f_1^{(n)}, \ldots, f_m^{(n)}$, то чтобы вычислить их суперпозицию нужно проделать следующее. Предположим, что каждая

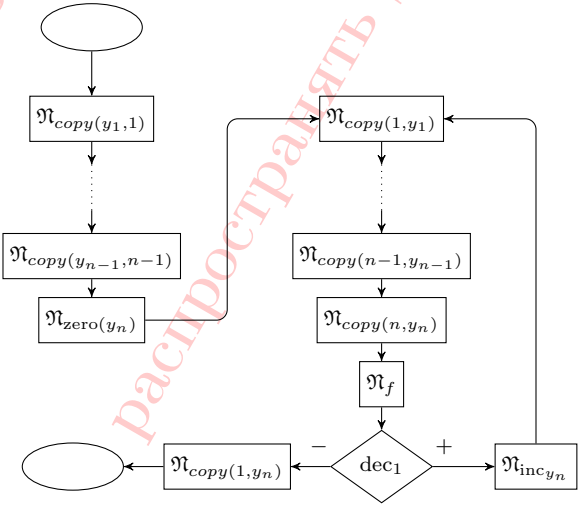
из перечисленных функций вычисляется некоторой счетчиковой машиной $\mathfrak{N}_g, \mathfrak{N}_{f_1}, \ldots, \mathfrak{N}_{f_m}$ с не более чем k счетчиками. Построим новую машину с $k + m + n$ счетчиками. Обозначим новые счетчики $y_1, \ldots, y_n$ и $z_1, \ldots, z_m$ соответственно. Тогда новая счетчиковая машина вычисляющая $h(\bar{x}) = g(f_1(\bar{x}), \ldots, f_m(\bar{x}))$ должна будет выполнить следующие действия:



Если вычислимы функции $f^{(n-1)}$ и $g^{(n-1)}$, то их примитивная рекурсия $h = \mathrm{Pr}[f, g]$ вычисляется так. Пусть функции f и g вычисляются счетчиковыми машинами $\mathfrak{N}_g$ и $\mathfrak{N}_f$ соответственно с не более чем k счетчиками. Построим новую машину с $k + n + 2$ счетчиками. Обозначим новые счетчики $y_1, \ldots, y_n, y_{n+1}$ и z. Счетчиковая машина для вычисления h может делать следующее:



Если вычислима функция f, то ее минимизация вычисляется так



□

Следствие 56. *Каждая частично рекурсивная функция вычислима на машине Тьюринга.*

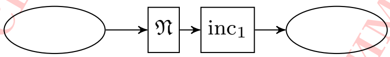
Мы видели, что функция Аккермана растет быстрее любой примитивно рекурсивной функции. Покажем, что аналогичным образом можно найти некоторую границу роста общерекурсивных функций.

Определение 50 (Функция «усердного бобра»). *Функция «усердного бобра» определяется так:* $\mathrm{bb}(x) = y$, *если существует функция* f, *которая вычислима на некоторой счетчиковой машине с* x *состояниями,* $f(0) = y$ *и* y *при этом является наибольшим возможным.*

Иначе говоря, $\mathrm{bb}(x) = y$, если y — самое большое число, которое может выдать машина с x состояниями при нулевом входе.

Следствие 57. *Функция* bb *является строго возрастающей.*

Доказательство. Если $\mathfrak{N}$ — машина с n состояниями, которая при нулевом аргументе дает наибольший результат — $\mathrm{bb}(n)$, то машина



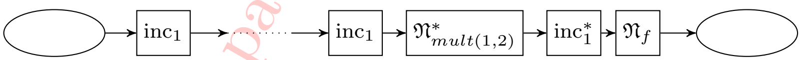
имеет $n+1$ состояние и дает результат бо́льший, чем машина $\mathfrak{N}$. Поэтому $\mathrm{bb}(n+1) > \mathrm{bb}(n)$. □

Теорема 58. *Функция* bb *растет быстрее любой о.р.ф.: для любой о.р.ф.* f *выполняется* $\mathrm{bb}(x) > f(x)$ *для всех* x, *начиная с некоторого.*

Доказательство. Предположим обратное, для некоторой о.р.ф. f существует бесконечно много x, для которых выполнено $\mathrm{bb}(x) \leqslant f(x)$.

Пусть $\mathfrak{N}_f$ — счетчиковая машина, вычисляющая функцию f. Предположим, что она имеет a состояний. Поскольку существует бесконечно много натуральных чисел x, для которых выполнено $\mathrm{bb}(x) \leqslant f(x)$, то среди них имеются и такие, которые больше $2a+16$. Зафиксируем одно из них. Тогда будет выполнено $a+7 < a+8 \leqslant \left[\frac{x}{2}\right]$.

Далее, можно построить машину $\mathfrak{N}^*_{mult(1,2)}$, которая удваивает значение первого счетчика и имеет 6 состояний (предложение 13 на с. 13). Рассмотрим следующую счетчиковую машину:



Здесь в начале идет $\left[\frac{x}{2}\right]$ команд инкремента, команду инкремента, помеченную звездочкой включаем, если число x является нечетным. Таким образом, функция, которую вычисляет эта машина на нулевом аргументе принимает значение $f(x) \geqslant \mathrm{bb}(x)$. Эта машина имеет не более $\left[\frac{x}{2}\right] + 6 + 1 +$

$a = 7 + \left[\frac{x}{2}\right] + a$ состояний. С другой стороны, $a + 7 < \left[\frac{x}{2}\right]$, поэтому

$$\left[\frac{x}{2}\right] + 6 + 1 + a = 7 + \left[\frac{x}{2}\right] + a < \left[\frac{x}{2}\right] + \left[\frac{x}{2}\right] \leqslant x.$$

Учитывая монотонность функции bb получаем, что

$$\mathrm{bb}(7 + \left[\frac{x}{2}\right] + a) < \mathrm{bb}(x).$$

Но $\mathrm{bb}(7 + \left[\frac{x}{2}\right] + a)$ — это наибольшее значение, которое может выдать счетчиковая машина с $7 + \left[\frac{x}{2}\right] + a$ состояниями на нулевом аргументе, поэтому $\mathrm{bb}(7 + \left[\frac{x}{2}\right] + a) \geqslant f(x)$, получаем противоречие:

$$\mathrm{bb}(x) \leqslant f(x) \leqslant \mathrm{bb}(7 + \left[\frac{x}{2}\right] + a) < \mathrm{bb}(x).$$

Значит наше предположение неверно. □

§ 1.3.5. Моделирование счетчиковой машины, универсальные функции

Определение 51 (Универсальная функция)**.** *Пусть K — некоторый класс m-местных функций. $m+1$-местная функция F называется универсальной в классе K, если выполняются следующие условия:*

1) *для любого числа x m-местная функция f_x такая, что $f_x(\bar{y}) = F(x, \bar{y})$ для всех $\bar{y}$, принадлежит K;*

2) *для любой функции f из K существует такое число x, что $f(\bar{y}) = F(x, \bar{y})$ для всех $\bar{y}$.*

Определение 52 (Универсальная ч.р.ф.)**.** *Если $m+1$-местная ч.р.ф. F является универсальной в классе всех m-местных ч.р.ф., то F называется универсальной ч.р.ф.*

Теорема 59. *Для каждого натурального $m > 0$ существует $m+1$-местная частично рекурсивная функция F_m, которая является универсальной в классе m-местных вычислимых (частичных) функций.*

Доказательство. Согласно следствию 28 на с. 35, мы можем считать, что любая m-местная вычислимая функция f вычислима на машине с $m+2$ счетчиками. Напомним, что в этом случае аргументы помещаются

в счетчики $1, \ldots, m$, а вычисленное значение располагаются в счетчике 1. Мы закодируем всевозможные $m+2$-счетчиковые машины натуральными числами и построим частично рекурсивную функцию $F_m(x, y_1, \ldots, y_m)$ так, что $F_m(x, y_1, \ldots, y_m) = z$ тогда и только тогда, когда счетчиковая машина с номером x останавливается на аргументах $y_1, \ldots, y_m$ и результат равен z.

Чтобы построить такое кодирование будем считать, что все состояния каждой $m+2$-счетчиковой машины $\mathfrak{N}$ пронумерованы натуральными числами: $q_0, \ldots, q_n$, причем q_0 — начальное состояние. Таким образом, команды машины обязательно имеют один из видов: $q_i \to \mathrm{inc}_k, q_j$ или $q_i \to \mathrm{dec}_k, q_j : q_{j'}$, где $k = 1, \ldots, m+2$.

Пусть p_i — i-ое простое число (нумерация начинается с нуля). Каждой команде вида $q_i \to \mathrm{inc}_k, q_j$ поставим в соответствие число

$$c(i) = p_i^{\langle 2k, \langle j,j \rangle \rangle},$$

а команде $q_i \to \mathrm{dec}_k, q_j : q_{j'}$ — число

$$c(i) = p_i^{\langle 2k+1, \langle j,j' \rangle \rangle}.$$

Отметим, что $c(i) \geqslant p_i > 1$ в обоих случаях. Всю машину закодируем как произведение

$$c(\mathfrak{N}) = \prod_{i=0}^{n} c(i).$$

Заметим, что при таком кодировании существуют:

1) примитивно рекурсивная функция $\mathrm{type}(x, i)$, определяющая инкремента (0) или декремента (1) команда для состояния q_i в машине с кодом x:
$$\mathrm{type}(x, i) = \langle \mathrm{ind}(x, \mathrm{iprime}(i)) \rangle_1 \bmod 2;$$

2) примитивно рекурсивная функция $\mathrm{counter}(x, i)$, определяющая к какому счетчику относится команда для состояния q_i в машине с кодом x:
$$\mathrm{counter}(x, i) = \left[\frac{\langle \mathrm{ind}(x, \mathrm{iprime}(i)) \rangle_1}{2} \right];$$

3) примитивно рекурсивные функции $\mathrm{go}_1(x, i)$ и $\mathrm{go}_2(x, i)$, определяющие переходы для состояния q_i в машине с кодом x:
$$\mathrm{go}_1(x, i) = \langle \langle \mathrm{ind}(x, \mathrm{iprime}(i)) \rangle_2 \rangle_1;$$

$$\mathrm{go}_2(x,i) = \langle\langle \mathrm{ind}(x, \mathrm{iprime}(i))\rangle_2\rangle_2 .$$

Если q_i является заключительным состоянием, то в программе не будет команды для q_i, следовательно, $\mathrm{ind}(x, \mathrm{iprime}(i))$ будет равно нулю и функция $\mathrm{counter}(x,i)$ тоже вернет нуль, чего не будет в противоположном случае.

В каждый момент времени конфигурация счетчиковой машины имеет вид $(q_i, a_1, \ldots, a_{m+2})$. Мы будем кодировать эту конфигурацию числом

$$2^i 3^{a_1} 5^{a_2} \cdots p_m^{a_m} p_{m+1}^{a_{m+1}} p_{m+2}^{a_{m+2}} .$$

Тогда по коду конфигурации v с помощью примитивно рекурсивных функций можно определить номер состояния:

$$\mathrm{state}(v) = \mathrm{ind}(v, 2),$$

и значение i-го счетчика:

$$\mathrm{value}(v,i) = \mathrm{ind}(v, \mathrm{iprime}(i)).$$

Начальная конфигурация тогда определяется $m+1$-местной примитивно рекурсивной функцией

$$\mathrm{begin}(x, y_1, \ldots, y_m) = 2^0 \times 3^{y_1} \times 5^{y_2} \times \cdots \times p_m^{a_m} \times p_{m+1}^0 \times p_{m+2}^0 = \\ = 3^{y_1} \times 5^{y_2} \times \cdots \times p_m^{a_m} .$$

Теперь определим $m+3$-местную примитивно рекурсивную функцию $\mathrm{next}(x, y_1, \ldots, y_m, t, v)$, которая по номеру счетчиковой машины x и коду конфигурации v будет возвращать код следующей конфигурации (аргументы $y_1, \ldots, y_m$ и t являются фиктивными, реально значение функции от них не зависит). Для удобства определим вспомогательную функцию $\mathrm{mlt}(x,v)$ — простое число, степенью которого кодируется значение счетчика для команды:

$$\mathrm{mlt}(x,v) = \mathrm{iprime}(\mathrm{counter}(x, \mathrm{state}(v))).$$

Тогда функцию $\mathrm{next}(x, y_1, \ldots, y_m, u, v)$ можно вычислить с использованием условной функции if. При $\mathrm{type}(x, \mathrm{ind}(v,2)) = 0$ ее значение равно

$$\frac{v}{2^{\mathrm{state}(v)}} \times 2^{\mathrm{go}_1(x, \mathrm{state}(v))} \times \mathrm{mlt}(x,v).$$

При $\mathrm{type}(x, \mathrm{state}(v)) = 1$ и $\mathrm{ind}(v, \mathrm{mlt}(x,v)) > 0$ ее значение равно

$$\frac{v}{2^{\mathrm{state}(v)} \times \mathrm{mlt}(x,v)} \times 2^{\mathrm{go}_1(x, \mathrm{state}(v))},$$

а в остальных случаях:

$$\frac{v}{2^{\mathrm{state}(v)}} \times 2^{\mathrm{go}_2(x,\mathrm{state}(v))}.$$

Таким образом, если число v кодирует конфигурацию счетчиковой машины с номером x в t-ый момент времени, то $\mathrm{next}(x, y_1, \ldots, y_m, t, v)$ будет кодировать конфигурацию этой же машины в $t+1$-ый момент.

Пусть $\mathrm{conf} = \Pr[\mathrm{begin}, \mathrm{next}]$. Тогда $\mathrm{conf}(x, y_1, \ldots, y_m, t)$ — конфигурация счетчиковой машины с программой x на входе $y_1, \ldots, y_m$ после выполнения t шагов.

Рассмотрим примитивно рекурсивную функцию

$$\mathrm{stop}(x, y_1, \ldots, y_m, t) = \mathrm{test}\,\mathrm{counter}(x, \mathrm{state}(\mathrm{conf}(x, y_1, \ldots, y_m, t))).$$

Как мы уже отметили выше, функция counter вернет номер счетчика, к которому относится команда на t-ом шаге вычисления, или нуль, если такой команды нет, то есть машина перешла в заключительное состояние. Таким образом, значением функции $\mathrm{stop}(x, y_1, \ldots, y_m, t)$ будет 0 или 1 в зависимости от того, перешла машина на t-ом шаге в заключительное состояние или нет. Пусть

$$\mathrm{quit} = \mu\,\mathrm{stop}\,.$$

Тогда результатом $\mathrm{quit}(x, y_1, \ldots, y_m)$ будет наименьший номер заключительной конфигурации. Если же машина зацикливается, то $\mathrm{quit}(x, y_1, \ldots, y_m)$ будет неопределено. Сама заключительная конфигурация может быть получена как

$$\mathrm{conf}(x, y_1, \ldots, y_m, \mathrm{quit}(x, y_1, \ldots, y_m)),$$

а значение первого счетчика в ней, то есть вычисленное значение:

$$\mathrm{value}(\mathrm{conf}(x, y_1, \ldots, y_m, \mathrm{quit}(x, y_1, \ldots, y_m)), 1).$$

Следовательно,

$$F_m(x, y_1, \ldots, y_m) = \mathrm{value}(\mathrm{conf}(x, y_1, \ldots, y_m, \mathrm{quit}(x, y_1, \ldots, y_m)), 1). \qquad \square$$

Следствие 60. *Каждая вычислимая (частичная) функция f является частично рекурсивной.*

Доказательство. По определению универсальной функции существует такое x_0, что $f(\bar{y}) = F(x_0, \bar{y})$ для всех $\bar{y}$. Если функция $F(x, \bar{y})$ является частично рекурсивной, то и $f(\bar{y}) = F(x_0, \bar{y})$ является. $\square$

Используя конструкцию из доказательства теоремы можно даже точнее указать вид этой ч.р.ф.

Предложение 61. *Для любой частично рекурсивной функции f существуют примитивно рекурсивные функции g и h такие, что $f = h(\mu g)$.*
Доказательство. Определим функцию g следующим образом:

$$g'(c, y_1, \ldots, y_m, u) = \mathrm{stop}(c, y_1, \ldots, y_m, t),$$

если $u = \langle c, y_1, \ldots, y_m, t\rangle$, или

$$g'(c, y_1, \ldots, y_m, u) = 1$$

в противном случае. Тогда значение функции $\mu g'$ на аргументах $c, y_1, \ldots, y_m$ будет равно $\langle c, y_1, \ldots, y_m, t\rangle$, если t — наименьший момент времени, когда состояние машины с номером c при работе на $y_1, \ldots, y_m$ оказалось заключительным, или же не будет определено вообще, если такого момента времени нет. Пусть

$$h(u) = \mathrm{value}(\mathrm{conf}(\langle u\rangle^1_{m+2}, \langle u\rangle^2_{m+2}, \ldots, \langle u\rangle^{m+1}_{m+2}, \langle u\rangle^{m+2}_{m+2}), 1).$$

Если c_0 — номер счетчиковой машины, которая вычисляет ч.р.ф. f, то $f = h(\mu g)$, где $g(\bar{y}, u) = g'(c_0, \bar{y}, u)$. □

Теорема 62. *Пусть счетчиковая машина (или машина Тьюринга) $\mathfrak{N}$ вычисляет общерекурсивную функцию $f(\bar{y})$ и для любых $\bar{y}$ время работы машины ограничено некоторой примитивно рекурсивной функцией $t(\bar{y})$. Тогда функция $f(\bar{y})$ также является примитивно рекурсивной.*
Доказательство. В доказательстве теоремы оператор минимизации применяется всего один раз и именно для того, чтобы найти номер шага, на котором счетчиковая машина остановится. Если этот номер мажорируется функцией $t(\bar{y})$ вместо полной минимизации можно использовать ограниченную:

$$\mathrm{quit}'_{\lim}(x, \bar{y}, t) = \mu_{\lim}\,\mathrm{stop}; \quad \mathrm{quit}_{\lim}(x, \bar{y}) = \mathrm{quit}'_{\lim}(x, \bar{y}, t(\bar{y})).$$

и тогда вся функция окажется примитивно рекурсивной согласно предложению 36 на с. 47. □

Теорема 63. *Не существует общерекурсивной двухместной функции, универсальной в классе одноместных общерекурсивных функций.*
Доказательство. Если бы существовала F_o — универсальная общерекурсивная функция, то $f(x) = F_o(x, x) + 1$ тоже была бы общерекурсивной функцией. Но тогда $f(y) = F_o(x_0, y)$ для некоторого x_0 для всех y. Подставляя $y = x_0$, получим $f(x_0) = F_o(x_0, x_0)$, а с другой стороны, $f(x_0) = F_o(x_0, x_0) + 1$ по определению. Противоречие. □

Теорема 64. *Существует общерекурсивная функция универсальная в классе m-местных примитивно рекурсивных функций. Ни одна из таких функций не является примитивно рекурсивной.*

Доказательство. Изменим определение функции stop из теоремы 59 на с. 65 следующим образом:

$$\operatorname{stop}(x,\bar{y}) = \mu_{v<A(\langle x\rangle_2,\max\bar{y})}\operatorname{go}_1(\langle x\rangle_1, \operatorname{state}(\operatorname{conf}(N,\bar{y},v))). \tag{1.9}$$

Здесь A — функция Аккермана, а с помощью $\max\bar{y}$ обозначен наибольший элемент набора $\bar{y}$. Рассмотрим

$$P(x,\bar{y}) = \operatorname{value}(\operatorname{conf}(x,\bar{y},\operatorname{stop}(x,\bar{y})),1).$$

Общерекурсивность этой функции очевидна. Покажем, что она является универсальной в классе примитивно рекурсивных функций.

Рассмотрим $f_x(\bar{y}) = P(x,\bar{y})$. Тогда в (1.9) ограничение в операторе минимизации имеет вид $v < A(\langle x\rangle_2, \max\bar{y})$. Поскольку $\langle x\rangle_2$ как и сам x зафиксирован, то $A(\langle x\rangle_2, \max\bar{y})$ является примитивно рекурсивной функцией (предложение 52 на с. 59). Но тогда и функция stop является примитивно рекурсивной (при фиксированном x), а значит и функция f_x.

Рассмотрим теперь произвольную m-местную примитивно рекурсивную функцию $f(\bar{y})$. Согласно следствию 28 на с. 35, она вычислима на $m+2$-счетчиковой машине $\mathfrak{N}$ и время ее вычисления $T(\bar{y})$ тоже является примитивно рекурсивной функцией. Следовательно, по теореме 49 на с. 55 существует n и $T(\bar{y}) \leqslant A(n,\max\bar{y})$ для всех $\bar{y}$. Пусть $c = c(\mathfrak{N})$ — код машины $\mathfrak{N}$, как он определен в доказательстве теоремы 59 на с. 65. Возьмем $x = \langle c,n\rangle$. Поскольку машина $\mathfrak{N}$ на входе $\bar{y}$ гарантированно останавливается за время не большее $A(\langle x\rangle_2, \max\bar{y})$, то функция $\operatorname{stop}(x,\bar{y})$ равняется номеру шага, на котором это произойдет, поэтому значение $P(x,\bar{y})$ будет равно возвращенному машиной $\mathfrak{N}$ значению, то есть $f(\bar{y})$.

Доказательство того, что универсальная функция не может быть примитивно рекурсивной аналогично доказательству теоремы 63 на предыдущей странице. □

Теорема 65. *Существуют частично рекурсивные функции, которые невозможно дополнить до общерекурсивных.*

Доказательство. Аналогично теореме 63 на предшествующей странице. Рассмотрим функцию $f(x) = F(x,x)+1$, которая тоже является частично рекурсивной. Предположим, что существует общерекурсивная функция f_o — дополнение f. Тогда $f_o(y) = F(x_0,y)$ для некоторого x_0 для всех y. Подставляя $y = x_0$, точно такими же рассуждениями получаем противоречие. □

§ 1.3.6. Нумерации, теорема о рекурсии

Предложение 66. *Существует двухместная примитивно рекурсивная функция* $\mathrm{inc}(i,x)$*, которая возвращает номер счетчиковой машины, увеличивающей на* x *значение счетчика* i*.*

Доказательство. Очевидно, что такая счетчиковая машина может выглядеть так:

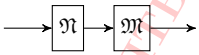
Тогда функция $\mathrm{inc}(i,x)$ легко строится по индукции с помощью примитивной рекурсии: $\mathrm{inc}(i,0)=1$, $\mathrm{inc}(i,x+1)=\mathrm{inc}(i,x)\times p_x^{\langle 2i,\langle x+1,x+1\rangle\rangle}$:

$$\mathrm{inc}=\mathrm{Pr}[1(\mathrm{id}_1^1),\mathrm{id}_3^3\times\mathrm{power}(\mathrm{iprime}(\mathrm{id}_3^2),\langle 2\times\mathrm{id}_3^1,\langle \mathrm{id}_3^2+1,\mathrm{id}_3^2+1\rangle\rangle)]$$

Здесь мы просто на каждом шаге добавляем к нашей счетчиковой машине команду вида

$$q_j\to \mathrm{inc}_i, q_{j+1}.$$ □

Предложение 67. *Существует двухместная примитивно рекурсивная функция, которая по* x *и* y *— номерам двух машин* $\mathfrak{N}$ *и* $\mathfrak{M}$ *соответственно — возвращает номер счетчиковой машины*



Мы будем обозначать этот номер с помощью $x\circ y$*.*

Доказательство. Напомним, что когда мы вычисляем номер счетчиковой машины, то считаем, что все состояния каким-то образом пронумерованы, и начальное состояние имеет номер 0. Пусть k — первое число, которое превосходит все номера состояний счетчиковой машины $\mathfrak{N}$. Чтобы построить требуемую машину мы должны сделать с программами исходных машин $\mathfrak{N}$ и $\mathfrak{M}$ следующие преобразования:

1) добавить к номерам всех состояний машины $\mathfrak{M}$ число k: q_0 заменить на q_k, q_1 на q_{k+1} и т.д., чтобы они не совпадали с номерами состояний машины $\mathfrak{N}$. Отметим, что начальным состоянием машины $\mathfrak{M}$ станет q_k;

2) заменить в программе машины $\mathfrak{N}$ все заключительные состояния на q_k;

3) соединить полученные две программы в одну.

Покажем, как выполнить каждое из указанных преобразований с помощью примитивно рекурсивных функций.

Сначала покажем, как построить функцию $k(u)$, которая по номеру счетчиковой машины возвращает наименьшее число, превосходящее номера всех состояний. Пусть

$$k^{(2)} = \Pr[0^{(1)}, \mathrm{if}(\mathrm{iprime}(\mathrm{id}_3^2) \mid \mathrm{id}_3^1, \mathrm{id}_3^2, \mathrm{id}_3^3)].$$

Таким образом, $k^{(2)}(u, v)$ — это номер наибольшего простого числа, из первых v простых чисел, которое делит u. Тогда $k = k^{(2)}(\mathrm{id}_1^1, \mathrm{id}_1^1)$.

Чтобы выполнить пункт 1) мы команды вида $q_j \to \mathrm{inc}_i, q_{j'}$ (или $q_j \to \mathrm{dec}_i, q_{j'} : q_{j''}$) в машине $\mathfrak{M}$ должны заменить на $q_{j+k(x)} \to \mathrm{inc}_i, q_{j'+k(x)}$ (или на $q_{j+k(x)} \to \mathrm{dec}_i, q_{j'+k(x)} : q_{j''+k(x)}$ соответственно) для всех $j = 0, \ldots, k(y) - 1$. Для номеров это соответствует замене каждого множителя вида $p_j^{\langle i, \langle j', j'' \rangle\rangle}$ на $p_{j+k(x)}^{\langle i, \langle j'+k(x), j''+k(x) \rangle\rangle}$:

$$f(x, y) = \prod_{j<k(y)} \mathrm{power}(\mathrm{iprime}(j + k(x)), \Big\langle \langle \mathrm{ind}(y, \mathrm{iprime}(j)) \rangle_2^1, \langle \mathrm{go}_1(y, j) + k(x), \mathrm{go}_2(y, j) + k(x) \rangle \Big\rangle).$$

Аналогично, для 2) мы в каждой команде любые заключительные состояния должны поменять на $k(x)$. Сама замена может быть выполнена при помощи функций if и counter

$$p(x, i) = \mathrm{if}(\mathrm{counter}(x, i), i, k(x)).$$

Тогда результат для всей программы можно получить так:

$$g(x) = \prod_{j<k(x)} \mathrm{power}(\mathrm{iprime}(j), \Big\langle \langle \mathrm{ind}(x, \mathrm{iprime}(j)) \rangle_2^1, \langle p(x, \mathrm{go}_1(x, j)), p(x, \mathrm{go}_2(x, j)) \rangle \Big\rangle).$$

После этого остается только перемножить полученные значения:

$$x \circ y = f(x, y) \times g(x). \qquad \square$$

Определение 53 (Нумерация функций)**.** *Нумерацией m-местных (частичных) функций называется $m+1$-местная (частичная) функция Φ (нумерующая функция). При этом x называется номером функции $f(\bar{y})$, если $\Phi(x, \bar{y}) = f(\bar{y})$ для всех $\bar{y}$. Функцию, которая имеет номер x в нумерации Φ также обозначают с помощью Φ_x.*

Нумерация называется вычислимой, если функция Φ является частично рекурсивной.

Нумерация называется главной, если для любой вычислимой нумерации Ψ существует общерекурсивная функция g такая, что $\Psi_x = \Phi_{g(x)}$ для всех x. В этом случае про функцию g говорят, что она сводит нумерацию Ψ к нумерации Φ.

Главная вычислимая нумерация называется геделевой.

Следствие 68. *Любая геделева нумерация m-местных функций Φ является универсальной ч.р.ф.*

Доказательство. Пусть Φ — произвольная геделева нумерация, по определению она вычислима, то есть является ч.р.ф. Пусть F — универсальная ч.р.ф. Тогда существует о.р.ф. g и для всех x выполнено $F_x = \Phi_{g(x)}$. Следовательно, для любой ч.р.ф. f получаем $f = F_x = \Phi_{g(x)}$ для некоторого x. □

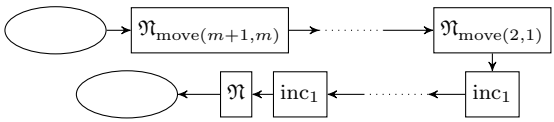
Предложение 69. *Универсальная частично рекурсивная функция F, построенная в в теореме 59 на с. 65, является геделевой нумерацией.*

Доказательство. Вычислимость F следует из построения. Пусть Φ — произвольная вычислимая нумерация m-местных (частичных) функций. Тогда функция Φ вычисляется некоторой счетчиковой машиной $\mathfrak{N}$. Возьмем h — ч.р.ф., которую вычисляет машина $\mathfrak{N}$.

Покажем, как по x — номеру функции f в нумерации Φ, построить номер f в нумерации F. Пусть $a_1, \ldots, a_m$ — номера машин $\mathfrak{N}_{\mathrm{move}(m+1,m)}$, $\ldots$, $\mathfrak{N}_{\mathrm{move}(2,1)}$ соответственно, c — номер машины $\mathfrak{N}$. Тогда рассмотрим п.р.ф.

$$g(x) = a_1 \circ \cdots \circ a_m \circ \mathrm{inc}(1, x) \circ c.$$

Очевидно, $g(x)$ будет номером такой машины:



По построению машина с номером $g(x)$ на аргументах $a_1, \ldots, a_m$ будет работать так же как машина $\mathfrak{N}$ на аргументах $x, a_1, \ldots, a_m$. То есть машина с номером $g(x)$ вычисляет функцию Φ_x, но тогда функция $F_{g(x)}$ равна Φ_x. □

Определение 54 (Экстенсиональная функция)**.** *Пусть Φ — нумерация ч.р.ф. Одноместная функция f называется э к с т е н с и о н а л ь н о й для нумерации Φ, если из $\Phi_x = \Phi_y$ следует, что $\Phi_{f(x)} = \Phi_{f(y)}$.*

Предложение 70. *Пусть о.р.ф. h является экстенсиональной для геделевой нумерации Φ. Тогда для любой геделевой нумерации Ψ существует «аналогичная» экстенсиональная о.р.ф. h', то есть из $\Phi_x = \Psi_y$ следует $\Phi_{h(x)} = \Psi_{h'(y)}$.*

Доказательство. Пусть о.р.ф. g сводит Ψ к Φ, а g' — Φ к Ψ. Возьмем $h'(z) = g'(h(g(z)))$.

Предположим, $\Phi_x = \Psi_y$ Тогда будет выполнено $\Phi_{g(y)} = \Psi_y$ по свойству сводящей функции, то есть $\Phi_{g(y)} = \Phi_x$. Далее получим $\Phi_{h(g(y)} = \Phi_{h(x)}$ по свойству функции h. Наконец, по свойству g' будем иметь $\Phi_{h(g(y))} = \Psi_{g'(h(g(y)))}$. Таким образом, $\Psi_{g'(h(g(y)))} = \Phi_{h(x)}$, то есть $\Psi_{h'(y)} = \Phi_{h(x)}$. □

Теорема 71 (Теорема об эффективной подстановке)**.** *Пусть Φ^n и Φ^m — геделевы нумерации n- и m-местных функций соответственно, $m \leqslant n$. Тогда существует о.р.ф. $\mathrm{S}_n^m(x, x_1, \ldots, x_{n-m})$ такая, что*

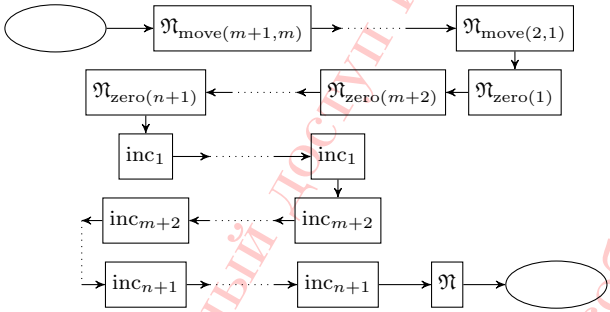
$$\Phi^m_{\mathrm{S}_n^m(c,x_1,\ldots,x_{n-m})}(y_1, \ldots, y_m) = \Phi^n_c(y_1, \ldots, y_m, x_1, \ldots, x_{n-m})$$

для любых $c, y_1, \ldots, y_m, x_1, \ldots, x_{n-m}$.

Доказательство. Сначала покажем это, когда F^m и F^n — это универсальные ч.р.ф., построенные по теореме 59 на с. 65 для моделирования счетчиковых машин. Пусть $a_m, \ldots, a_1$ — номера счетчиковых машин $\mathfrak{N}_{\mathrm{move}(m+1,m)}, \ldots, \mathfrak{N}_{\mathrm{move}(2,1)}$, b — номер счетчиковой машины $\mathfrak{N}$, вычисляющей $n+1$-местную функцию F^n, $d_1, d_{m+2}, \ldots, d_{n+1}$ — номера счетчиковых машин $\mathfrak{N}_{\mathrm{zero}(1)}, \mathfrak{N}_{\mathrm{zero}(m+2)}, \ldots, \mathfrak{N}_{\mathrm{zero}(n+1)}$ соответственно. Тогда

$$\begin{aligned}\mathrm{S}_n^m(c, x_1, \ldots, x_{n-m}) = a_m \circ \cdots \circ a_1 \circ d_1 \circ d_{m+2} \circ \cdots \circ d_{n+1} \circ \\ \circ \mathrm{inc}(1, c) \circ \mathrm{inc}(m+2, x_1) \circ \cdots \circ \mathrm{inc}(n+1, x_{n-m}) \circ b\end{aligned}$$

это номер следующей счетчиковой машины $\mathfrak{M}_{s_n^m(c,x_1,\ldots,x_{n-m})}$:



Если в начальный момент времени в первых m счетчиках машины $\mathfrak{M}_{\mathrm{S}_n^m(c,x_1,\ldots,x_{n-m})}$ находились числа $y_1,\ldots,y_m$, то к началу работы машины $\mathfrak{N}$ значения первых $n+1$ счетчиков будут такими $c, y_1,\ldots,y_m,x_1,\ldots,x_{n-m}$. Следовательно, результатом работы машины $\mathfrak{M}_{\mathrm{S}_n^m(c,x_1,\ldots,x_{n-m})}$ будет

$$F^n(c,y_1,\ldots,y_m,x_1,\ldots,x_{n-m}) = F_c^n(y_1,\ldots,y_m,x_1,\ldots,x_{n-m}).$$

Если же Φ^n и Φ^m — произвольные геделевы нумерации, то существуют о.р.ф. f, сводящая F^m к Φ^m, и g, сводящая Φ^n к F^n. Тогда нужная функция получается как суперпозиция: $f(\mathrm{S}_n^m(g))$. □

Следствие 72. *Для любых n-местной ч.р.ф. f, $m \leqslant n$ и геделевой нумерации m-местных функций Φ существует о.р.ф. S_f^m такая, что для любых $y_1,\ldots,y_m,x_1,\ldots,x_{n-m}$ имеет место*

$$\Phi_{\mathrm{S}_f^m(x_1,\ldots,x_{n-m})}(y_1,\ldots,y_m) = f(y_1,\ldots,y_m,x_1,\ldots,x_{n-m})$$

Доказательство. Пусть Ψ — геделева нумерация n-местных функций, c — номер f в этой нумерации. Тогда очевидно, $\mathrm{S}_f^m(x_1,\ldots,x_{n-m}) = \mathrm{S}_n^m(c,x_1,\ldots,x_{n-m})$. □

Используя эту теорему можно придать более конструктивный вид предложению 61 на с. 69.

Предложение 73. *Пусть Φ и Ψ — геделевы нумерации m и $m+1$-местных ч.р.ф. соответственно. Тогда существуют одноместные о.р.ф. h и g такие, что*

1) *$\Psi_{g(x)}$ является о.р.ф. для любого x и $\operatorname{rng}\Psi_{g(x)} \subseteq \{0;1\}$;*

2) *$\Phi_x = h(\mu\Psi_{g(x)})$ для любого x.*

Доказательство. Напомним, что согласно предложению 61 на с. 69, универсальная ч.р.ф. F может быть представлена в виде $h(\mu g')$ для одноместной п.р.ф. h и $m+2$-местной п.р.ф. g'. Согласно следствию 72 на предыдущей странице существует о.р.ф. $\mathrm{S}_{g'}^{m+1}$, для которой

$$\Psi_{\mathrm{S}_{g'}^{m+1}(c)}(\bar{y},u) = g'(c,\bar{y},u).$$

Пусть о.р.ф. f сводит Φ к F, $f(d)$ — номер функции Φ_d в нумерации F, то есть номер счетчиковой машины, вычисляющей Φ_d. Тогда $g = \mathrm{S}_{g'}^{m+1}(f)$.

В самом деле,

$$\Psi_{g(x)}(\bar{y},u) = \Psi_{\mathrm{S}_{g'}^{m+1}(f(d))}(\bar{y},u) = g'(f(d),\bar{y},u).$$

Поскольку g' — п.р.ф., а f — о.р.ф., то $\Psi_{g(x)}$ — о.р.ф. Далее, g' определена так, что принимает лишь значения 0 и 1. Наконец,

$$\Phi_x(\bar{y}) = F_{f(x)}(\bar{y}) = h((\mu g')(f(x),\bar{y})) = h((\mu\Psi_{g(x)})(\bar{y})),$$

что и требуется. □

Теорема 74 (Теорема о неподвижной точке)**.** *Пусть Φ — геделева нумерация одноместных ч.р.ф. Для всякой общерекурсивной функции h существует число x_0 такое, что $\Phi_{f(x_0)} = \Phi_{x_0}$. Иначе говоря, если рассматривать f как функцию преобразования программ, получаем, что f переводит некоторую программу в эквивалентную.*

Доказательство. Рассмотрим двухместную ч.р.ф. $g = \Phi(\Phi(\mathrm{id}_2^2,\mathrm{id}_2^2),\mathrm{id}_2^1)$. Согласно следствию 72 на предшествующей странице существует о.р.ф. S_g^1, для которой

$$\Phi_{\mathrm{S}_g^1(x)}(y) = g(y,x) = \Phi(\Phi(x,x),y).$$

Функция $h(\mathrm{S}_g^1)$ тоже является общерекурсивной, поэтому она имеет некоторый номер n в нумерации Φ: $\Phi_n(x) = h(\mathrm{S}_g^1(x))$. Положим $x_0 = \mathrm{S}_g^1(n)$. Тогда получим

$$\Phi_{h(x_0)}(y) = \Phi(h(x_0),y) = \Phi(h(\mathrm{S}_g^1(n)),y) = \Phi(\Phi_n(n),y) =$$
$$= \Phi(\Phi(n,n),y) = g(y,n) = \Phi_{\mathrm{S}_g^1(n)}(y) = \Phi_{x_0}(y).$$

□

Следствие 75. *Для любой геделевой нумерации Φ существует о.р.ф. $h = \Phi_n$ такая, что значение $h(x) = n$ для всех x.*

Доказательство. Рассмотрим двухместную функцию $g = \mathrm{id}_2^2$: $g(y, x) = x$. По следствию 72 на с. 75 существует о.р.ф. S_g^1, для которой

$$\Phi_{\mathrm{S}_g^1(x)}(y) = g(y, x) = x.$$

Применим теорему о неподвижной точке к о.р.ф. S_g^1. Пусть n — неподвижная точка: $\Phi_n = \Phi_{\mathrm{S}_g^1(n)}$. Получим

$$\Phi_n(y) = \Phi_{\mathrm{S}_g^1(n)}(y) = g(y, n) = n.$$

Следовательно, $h = \Phi_n$ — требуемая функция. □

Теорему о неподвижной точке можно легко распространить на функции произвольного числа аргументов.

Теорема 76. *Пусть Φ — геделева нумерация m-местных ч.р.ф. Для всякой $n+1$-местной о.р.ф. h существует n-местная о.р.ф. f такая, что $\Phi_{h(f(\bar{x}),\bar{x})} = \Phi_{f(\bar{x})}$ для всех $\bar{x}$.*

Доказательство. Пусть Ψ — геделева нумерация $n+1$-местных функций. Согласно следствию 72 на с. 75 существует о.р.ф. S^m, для которой

$$\Phi_{\mathrm{S}^m(z,\bar{x})}(\bar{y}) = \Phi(\Psi(z, z, \bar{x}), \bar{y}).$$

Функция $h(\mathrm{S}^m(z, \bar{x}), \bar{x})$ тоже является общерекурсивной, поэтому она имеет некоторый номер k в нумерации Ψ: $\Psi_k(z, \bar{x}) = h(\mathrm{S}^m(z, \bar{x}), \bar{x})$. Положим $f(\bar{x}) = \mathrm{S}^m(k, \bar{x})$. Тогда получим

$$\begin{aligned}\Phi_{f(\bar{x})}(\bar{y}) = \Phi_{\mathrm{S}^m(k,\bar{x})}(\bar{y}) = \Phi(\Psi(k, k, \bar{x}), \bar{y}) = \Phi(\Psi_k(k, \bar{x}), \bar{y}) = \\ = \Phi(h(\mathrm{S}^m(k, \bar{x}), \bar{x}), \bar{y}) = \Phi(h(f(\bar{x}), \bar{x}), \bar{y}) = \Phi_{h(f(\bar{x}),\bar{x})}(\bar{y})\end{aligned}$$

□

Глава 2

Разрешимые и неразрешимые проблемы

Глава 3

Логика высказываний

Глава 4

Логика предикатов

Ответы и решения