МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. ЛОМОНОСОВА

Факультет вычислительной математики и кибернетики

С. С. Марченков

ИЗБРАННЫЕ ГЛАВЫ ДИСКРЕТНОЙ МАТЕМАТИКИ

Учебное пособие

Москва – 2015

УДК 519.71(075.8)
ББК

Печатается по решению редакционно-издательского совета факультета Вычислительной математики и кибернетики МГУ имени М.В. Ломоносова

Р е ц е н з е н т ы :
В.А. Захаров, д.ф.-м.н., профессор;
Д.С. Романов, к.ф.-м.н., доцент

Марченков С.С.

Избранные главы дискретной математики: Учебное пособие. — М.: Издательский отдел факультета ВМиК МГУ им. М.В. Ломоносова (лицензия ИД № 05899 от 24.09.2001); МАКС Пресс, 2015. — 134 с.
ISBN
ISBN

Пособие написано на основе потокового курса «Дополнительные главы дискретной математике», который автор на протяжении ряда лет читает на факультете вычислительной математики и кибернетики МГУ. Пособие состоит их четырех глав. В главе 1 «Функции многозначной логики» даются первоначальные сведения по функциям многозначной логики. Часть из них используется в главах 2 и 3. Глава 2 «Конечные автоматы-распознаватели» посвящена описанию конечно-автоматных множеств в терминах конечных автоматов, с использованием правоинвариантного отношения эквивалентности и на основе регулярных выражений. В главе 3 «Конечные автоматы-преобразователи» рассматривается класс конечно-автоматных функций, определенных на бесконечных последовательностях. Доказывается конечная порождаемость этого класса относительно операций суперпозиции и обратной связи и невозможность обойтись только операцией суперпозиции. В главе 4 «Машины Тьюринга и вычислимые функции» определяются машины Тьюринга и функции, вычислимые на машинах Тьюринга. Доказывается совпадение данного класса функций с классом частично рекурсивных функций. Вводятся понятия P-сводимости и NP-полноты. Устанавливается существание NP-полных проблем. Каждый параграф снабжен задачами и упражнениями.

Для студентов, аспирантов и научных сотрудников, специализирующихся в области дискретной математики и кибернетики.

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

В курсах дискретной математики, читаемых в российских университетах (а также в значительном части технических вузов), сложился определенный круг обязательных разделов, без которых изложение дискретной математики не может считаться полноценным. К таким разделам следует в первую очередь отнести разделы, посвященные булевым функциям и их обобщениям — функциям многозначной логики, конечным автоматам, машинам Тьюринга и вычислимым функциям. В различных курсах дискретной математики данные разделы могут быть представлены по-разному: это касается не только объема материала, включенного в разделы, но и степени подробности и проработки входящих в них тем.

В Московском государственном университете обязательный курс дискретной математики впервые был прочитан С.В. Яблонским на факультете вычислительной математики и кибернетики в 1971–72 учебном году. В 1979 г. появилось первое издание учебника С.В. Яблонского «Введение в дискретную математику» [8], а двумя годами ранее — первое издание сборника «Задачи и упражнения по дискретной математике» [3] Г.П. Гаврилова и А.А. Сапоженко. Впоследствии эти книги несколько раз переиздавались и переводились на другие языки. Не будет преувеличением сказать, что они внесли значительный вклад в становление и развитие дискретной математики в нашей стране.

На факультете вычислительной математики и кибернетики МГУ курс дискретной математики разбит на две части: первая — базовая — читается всем потокам первого курса; вторая — «Дополнительные главы дискретной математики» — в пятом семестре на втором потоке. И если базовая часть не претерпела существенных изменений по сравнению с курсом (и учебником) С.В. Яблонского (современное изложение этой части имеется в книге [1] В.Б. Алексеева), то во второй части произошли значительные изменения. Появилась потребность в изложении этой части в виде отдельного учебного пособия.

В настоящем учебном пособии представлены три раздела дискретной математики: функции многозначной логики, конечные автоматы, машины Тьюринга и вычислимые функции. Последние два раздела составляют курс «Дополнительные главы дискретной математики», который автор на протяжении ряда лет читает на факультете вычислительной математики и кибернетики. Раздел «Функции многозначной логики» включен в пособие, чтобы обеспечить полноту и независимость доказательств

некоторых утверждений для конечных автоматов. В пособие не включена глава по булевым функциям, поскольку в настоящее время имеется несколько доступных книг (см., например, [8, 1, 6]), в которых эта тема изложена достаточно подробно.

Раздел «Конечные автоматы» в пособии разбит на две главы: «Конечные автоматы-распознаватели» и «Конечные автоматы-преобразователи». В первой из этих глав содержится материал, который сравнительно редко встречается в учебниках по дискретной математике. Здесь собраны результаты, касающиеся различных определений конечно-автоматных множеств: собственно через конечные автоматы, с помощью правоинвариантного отношения эквивалентности и через регулярные выражения Клини. Приведены теоремы, показывающие эквивалентность всех определений.

Другая глава, посвященная конечным автоматам, содержит более традиционный материал, который на протяжении многих лет входит в университетские курсы дискретной математики. Единственная особенность здесь состоит в том, что в пособии довольно подробно рассмотрена связь между конечно-автоматными функциями, определенными над конечными слова и над бесконечными последовательностями (сверхсловами).

Большая часть главы «Машины Тьюринга и вычислимые функции» составлена из результатов, постоянно входящих в подобные курсы дискретной математики. Здесь определены класс функций, вычислимых на машинах Тьюринга, класс частично рекурсивных функций и доказано совпадение этих классов. Кроме того, имеются параграфы, посвященные понятиям P-сводимости и NP-полноты. В частности, приведено доказательство существования NP-полных проблем (теорема С. Кука).

Пособие содержит большое количество задач и упражнений, к которым даны ответы и решения.

Автор признателен Е.А. Колмакову за помощь в оформлении рисунков.

Глава 1

ФУНКЦИИ МНОГОЗНАЧНОЙ ЛОГИКИ

§ 1. Основные понятия

Функции многозначной логики представляют собой обобщение булевых функций. Поэтому многие вопросы для функций многозначной логики рассматриваются и решаются по аналогии с булевыми функциями. Однако существует ряд особенностей, которые характерны только для функций многозначной логики (начиная с функций трехзначной логики) и которых нет у булевых функций. В связи с этим теория функций многозначной логики значительно богаче и сложнее теории булевых функций.

Введем ряд определений. Пусть k — натуральное число, $k \geqslant 2$ и $E_k = \{0, 1, \ldots, k-1\}$. Будем рассматривать функции на множестве E_k, т.е. функции, отображающие декартовы степени E_k в E_k. Множество всех таких функций обозначим через P_k. Функции из P_k называются *функциями k-значной логики* (при $k = 2$ получаем *булевы функции* или *функции алгебры логики*). Если $Q \subseteq P_k$ и n — натуральное число, то пусть $Q^{(n)}$ обозначает множество всех функций из Q, заданных на E_k^n (множество n-местных функций из Q).

Обычно при определении n-местной функции f из P_k значение функции f на (упорядоченном) наборе $(a_1, \ldots, a_n)$ из E_k^n записывают в виде $f(a_1, \ldots, a_n)$. В связи с этим n-местную функциию f принято назвать функцией от n переменных и изображать в виде $f(x_1, \ldots, x_n)$, где $(x_1, \ldots, x_n)$ — упорядоченная n-ка переменных. При этом предполагается, что каждая переменная x_i $(1 \leqslant i \leqslant n)$ принимает значения из i-го сомножителя декартовой степени E_k^n. В частности, в наборе $(a_1, \ldots, a_n)$ переменная x_i принимает значение a_i.

Нетрудно подсчитать число функций в множестве $P_k^{(n)}$. В самом деле, всякая функция из $P_k^{(n)}$ определена на множестве E_k^n, состоящем из k^n элементов, и принимает значения в множестве E_k, имеющем k элементов. Таким образом, задать n-местную функцию из P_k значит приписать каждому набору из E_k^n одно из k возможных значений. Это можно сделать ровно k^{k^n} способами.

Для любого $n \geqslant 1$ и любого i $(1 \leqslant i \leqslant n)$ обозначим через $e_i^n(x_1, \ldots, x_i, \ldots, x_n)$ *селекторную функцию* из P_k, значения которой совпадают со значениями переменной x_i. Иногда, если это не приводит к недоразумению, вместо функции $e_i^n(x_1, \ldots, x_i, \ldots, x_n)$ будем писать просто переменную x_i.

Говорят, что функция $f(x_1, \ldots, x_i, \ldots, x_n)$ *существенно зависит* от переменной x_i, если найдутся такие значения $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n$ переменных $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$, что функция одной переменной $f(a_1, \ldots, a_{i-1}, x_i, a_{i+1}, \ldots, x_n)$ отлична от константы. Если функция $f(x_1, \ldots, x_n)$ существенно зависит от переменной x_i, то переменная x_i называется *существенной переменной* функции $f(x_1, \ldots, x_n)$. В противном случае переменная x_i называется *несущественной* или *фиктивной переменной* функции $f(x_1, \ldots, x_n)$.

Из определения существенной зависимости видно, что при вычислении значений функции реально используются лишь значения существенных переменных. В связи с этим возникает желание освободиться от «ненужных» фиктивных переменных. Если, например, известно, что существенными переменными функции $f(x_1, \ldots, x_n)$ являются только переменные $x_1, \ldots, x_m$, то избавиться в функции $f(x_1, \ldots, x_n)$ от фиктивных переменных $x_{m+1}, \ldots, x_n$ можно по-разному. Можно, например, значения функции f рассматривать только на наборах $(a_1, \ldots, a_n)$, у которых $a_{m+1} = \ldots = a_n = c$, где $c \in E_k$. Это соответствует подстановке константы c на места всех переменных $x_{m+1}, \ldots, x_n$: $f(x_1, \ldots, x_m, c, \ldots, c)$. Другой способ состоит в том, чтобы придать всем переменным $x_{m+1}, \ldots, x_n$ значения одной из переменных $x_1, \ldots, x_m$, например, переменной x_1. В таких случаях говорят, что переменные $x_{m+1}, \ldots, x_n$ *отождествляются* с переменной x_1: $f(x_1, \ldots, x_m, x_1, \ldots, x_1)$.

На практике часто возникает потребность «добавить» к функции $g(x_1, \ldots, x_m)$ фиктивные переменные $x_{m+1}, \ldots, x_n$. Для этого искомую функцию $f(x_1, \ldots, x_n)$, как правило, определяют равенством

$$f(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n) = g(x_1, \ldots, x_m),$$

которое считается верным при любых значениях переменных $x_1, \ldots, x_n$. Однако с алгебраической и функциональной точек зрения более оправданным представляется использование селекторных функций. Именно, переход от функции g к функции f выполняется с помощью «регулярной» подстановки селекторных функций в функцию g:

$$f(x_1, \ldots, x_m, x_{m+1}, \ldots, x_n) = g(e_1^n(x_1, \ldots, x_n), \ldots, e_m^n(x_1 \ldots, x_n)).$$

Пусть Q — непустое множество функций из P_k. Введем понятие *формулы над* Q. Одновременно всякой формуле над Q будет сопоставлена функция из P_k, которая реализуется этой формулой.

Предположим, что все функции из Q имеют индивидуальные обозначения. Если символом f обозначена n-местная функция из Q, а $x_1, \ldots, x_n$ суть символы переменных, то выражение $f(x_1, \ldots, x_n)$ называется *формулой над* Q. Формуле $f(x_1, \ldots, x_n)$ сопоставляем функцию из Q, имеющую обозначение f. Говорим, что эта функция *реализуется формулой* $f(x_1, \ldots, x_n)$.

Далее, если символом g обозначена функция из Q от m переменных, а $\Phi_1, \ldots, \Phi_m$ — либо формулы над Q, либо символы переменных (не обязательно различные), то выражение $g(\Phi_1, \ldots, \Phi_m)$ есть *формула над* Q. Предположим, что выражениям Φ_i, отличным от символов переменных, уже сопоставлены функции f_i. Если выражение Φ_i представляет собой символ переменной x_j, то сопоставим ему функцию $f_i(x_j)$, равную переменной x_j. Тогда формуле $g(\Phi_1, \ldots, \Phi_m)$ сопоставим функцию $g(f_1, \ldots, f_m)$, *реализуемую этой формулой.*

(В этом месте необходимо пояснить, что представляет собой функция, которая реализуется, например, формулой

$$g(f_1(x_1^1, \ldots, x_{n_1}^1), \ldots, f_m(x_1^m, \ldots, x_{n_m}^m)), \tag{1.1}$$

где функции $g, f_1, \ldots, f_m$ считаются известными, а переменные $x_1^1, \ldots, x_{n_m}^m$ не предполагаются различными. Будем считать, что переменные $x_1^1, \ldots, x_{n_m}^m$ выбраны из последовательности $x_1, x_2, \ldots$ (в иных случаях множество переменных $x_1^1, \ldots, x_{n_m}^m$ считаем линейно упорядоченным). Пусть, например, это переменные $x_{i_1}, \ldots, x_{i_l}$, где $i_1 < \ldots < i_l$. Тогда формула (1.1) реализует функцию $h(x_{i_1}, \ldots, x_{i_l})$, значения которой вычисляются следующим образом. Для произвольного набора $(a_1, \ldots, a_l)$ из E_k^l (значение a_j соответствует переменной x_{i_j}) образуется m наборов $\tilde{a}_1, \ldots, \tilde{a}_m$, которые отвечают наборам переменных $(x_1^1, \ldots, x_{n_1}^1), \ldots, (x_1^m, \ldots, x_{n_m}^m)$. Затем к наборам $\tilde{a}_1, \ldots, \tilde{a}_m$ применяются соответственно функции $f_1, \ldots, f_m$. Получается набор значений $(b_1, \ldots, b_m)$, к которому, в свою очередь, применяется функция g.)

Если функция f реализуется формулой, в состав которой помимо символов переменных входят только символы функций $f_1, \ldots, f_s$, то говорят, что функция f является (или образована) *суперпозицией функций* $f_1, \ldots, f_s$. Вообще, под операцией *суперпозиции* понимают (частичную) операцию, которая, используя формульную выразимость одной функции

через другие, позволяет по конечному набору функций F определять новую функцию, реализуемую формулой над F.

Важным частным случаем операции суперпозиции является отождествление переменных. Говорят, что функция $g(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n)$ получена из функции $f(x_1, \ldots, x_n)$ *отождествлением i-й и j-й переменных* $(1 \leqslant i < j \leqslant n)$, если

$$g(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n) = f(x_1, \ldots, x_{j-1}, x_i, x_{j+1}, \ldots, x_n).$$

Более сложные случаи отождествления нескольких переменных получаются последовательным выполнением отождествлений двух переменных.

Как видно из приведенных выше определений, введение операции суперпозиции на основе понятия формулы содержит некоторые трудности. Прежде всего, это касается определения функций, реализуемых формулами вида (1.1). Было бы гораздо удобнее, например, рассматривать формулы вида

$$g(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)).$$

Однако при этом есть риск потерять в общности: функции $f_1, \ldots, f_m$ вовсе не обязаны зависеть от одних и тех же переменных.

Существует по меньшей мере два пути выхода из этого положения. Можно считать, что наряду с каждой функцией у нас имеются все функции, полученные из нее введением несущественных переменных. Другой подход заключается в том, чтобы изначально (например, при определении замыкания) рассматривать только системы функций, включающие все селекторные функции. Второй подход представляется более «алгебраическим». Он не содержит некоторых противоречий, присущих первому подходу, однако требует наличия множества «вспомогательных» селекторных функций.

Пусть $Q \subseteq P_k$. *Замыканием* множества функций Q называтся множество всех функций, которые можно реализовать формулами над Q. Иными словами, замыкание множества Q состоит из всех функций, которые являются суперпозициями функций из Q. Замыкание множества Q будем обозначать через $[Q]$. Множество функций Q назовем (*функционально*) *замкнутым*, если $[Q] = Q$. Замкнутые множества принято также называть *замкнутыми классами*.

Нетрудно убедиться, что операция замыкания обладает следующими свойствами (далее Q, R — произвольные множества функций из P_k):

1. $Q \subseteq [Q]$;

2. Если $Q \subseteq R$, то $[Q] \subseteq [R]$;
3. $[[Q]] = [Q]$.

Пусть Q — замкнутый класс и $R \subseteq Q$. Говорят, что R является *порождающей системой* в классе Q (или R *порождает* Q, или R *полна* в Q), если $[R] = Q$. Замкнутый класс Q называется *конечно порождаемым*, если он имеет конечную порождающую систему. *Базисом* замкнутого класса называется такая порождающая его система, которая перестает быть порождающей системой после удаления любой принадлежащей ей функции.

УПРАЖНЕНИЯ

1. Подсчитать число функций в множестве $P_k^{(n)}$, существенно зависящих от всех переменных.

§ 2. Стандартные полные системы

Пусть $k \geqslant 2$. Для любого $i \in E_k$ положим

$$J_i(x) = \begin{cases} k-1, & \text{если } x = i, \\ 0 & \text{в противном случае.} \end{cases}$$

Наряду с функциями $J_0, \ldots, J_{k-1}$ в теореме 1.1 рассматриваются следующие функции из P_k: константы $0, 1, \ldots, k-1$ (их можно считать функциями от одной переменной), минимум $\min(x_1, x_2)$ и максимум $\max(x_1, x_2)$. Функции min и max от от нескольких переменных получаются из соответствующих двуместных функций очевидными суперпозициями.

Теорема 1.1 (о разложении функции по первой переменной). *Для любой функции $f(x_1, \ldots, x_n)$ из P_k справедливо представление*

$$f(x_1, \ldots, x_n) = \max\{\min(J_0(x_1), f(0, x_2, \ldots, x_n)), \ldots \\ \ldots, \min(J_{k-1}(x_1), f(k-1, x_2, \ldots, x_n))\}. \qquad (1.2)$$

Доказательство. Придавая переменной x_1 последовательно значения $0, 1, \ldots, k-1$, непосредственно убеждаемся в справедливости соотношения (1.2).

Следствие 1 (о разложении функции по всем переменным). *Для любой функции $f(x_1, \ldots, x_n)$ из P_k имеет место представление*

$$f(x_1,\dots,x_n)=\max\{\min(J_0(x_1),\dots,J_0(x_n),f(0,\dots,0)),$$
$$\min(J_0(x_1),\dots,J_0(x_{n-1}),J_1(x_n),f(0,\dots,0,1)),\dots$$
$$\dots,\min(J_{k-1}(x_1),\dots,J_{k-1}(x_n),f(k-1,\dots,k-1))\}. \quad (1.3)$$

Доказательство. Применим разложение (1.2) по переменной x_2 к функциям $f(0,x_2,\dots,x_n),\dots,f(k-1,x_2,\dots,x_n)$, затем — по переменной x_3 к образовавшимся функциям

$$f(0,0,x_3,\dots,x_n),f(0,1,x_3,\dots,x_n),\dots,f(k-1,k-1,x_3,\dots,x_n)$$

и т.д. При этом используем легко проверяемое тождество

$$\min(x,\max(y,z))=\max(\min(x,y),\min(x,z)).$$

Соотношение (1.3) можно проверить и непосредственно, подставляя в левую и правую части (1.3) последовательно все k^n наборов из E_k^n.

Следствие 2. *При любом $k\geqslant 2$ система функций*

$$0,\ 1,\ \dots,\ k-1,\ J_0(x),\ \dots,\ J_{k-1}(x),\ \min(x_1,x_2),\ \max(x_1,x_2) \quad (1.4)$$

полна в классе P_k.

Формулу (1.3) можно записать более компактно, если функцию max обозначить через $\vee$, а функцию min — через &:

$$f(x_1,\dots,x_n)=\bigvee_{(\sigma_1,\dots,\sigma_n)\in E_k^n} J_{\sigma_1}(x_1)\,\&\,\dots\,\&\,J_{\sigma_n}(x_n)\,\&\,f(\sigma_1,\dots,\sigma_n). \quad (1.5)$$

Представление (1.5) есть обобщение хорошо известного представления булевых функций в виде совершенной дизъюнктивной нормальной формы. При этом функцию f можно считать отличной от тождественного нуля, а в формуле (1.5) следует опустить дизъюнктивные слагаемые, в состав которых входит сомножители $f(\sigma_1,\dots,\sigma_n)$, равные 0.

Обозначим через $\bar{x}$ *отрицание Поста* — функцию $x+1$, где сложение рассматривается по модулю k. Отметим, что при $k=2$ отрицание Поста совпадает с булевым отрицанием $\bar{x}$.

Следствие 3. *При любом $k\geqslant 2$ система функций $\{\bar{x},\max(x_1,x_2)\}$ полна в классе P_k.*

Доказательство. Покажем, как суперпозициями функций системы $\{\bar{x},\max(x_1,x_2)\}$ получить все функции системы (1.4).

Прежде всего, суперпозициями функции $\bar{x}$ образуем функции $x+2,\ x+3,\ldots,x+k-1$ (все сложения проводятся по модулю k) и проверяем, что

$$\max(x, x+1, \ldots, x+k-1) = k-1.$$

Из константы $k-1$ с помощью функции $\bar{x}$ получаем все остальные константы: $0=(k-1)+1,\ 1=0+1,\ldots$. Далее проверяем, что

$$J_0(x) = \max(x, x+1, \ldots, x+k-2)+1.$$

В самом деле, если $x=0$, то $\max(0,1,\ldots,k-2)+1=(k-2)+1=k-1$. Если же $x\neq 0$, то $x+(k-1-x)=k-1$ и потому $\max(x, x+1, \ldots, x+k-2)+1=(k-1)+1=0$ (по модулю k).

Теперь при $i\neq 0$ получаем $J_i(x)=J_0(x+k-i)$. Если же $0\leqslant i\leqslant k-2$, то пусть

$$g_i(x) = \begin{cases} k-1-i, & \text{если } x=i, \\ 0 & \text{в противном случае} \end{cases}$$

(отметим, что функция g_0 совпадает с функцией J_0). Легко убедиться в том, что имеет место тождество

$$g_i(x) = k-i+\max(J_i(x), i).$$

Обозначив через $\sim x$ функцию $k-1-x$ (*отрицание Лукасевича*), получаем

$$\sim x = \max(g_0(x), \ldots, g_{k-2}(x)).$$

Остается заметить, что

$$\min(x_1, x_2) = \sim \max(\sim x_1, \sim x_2).$$

Следствие доказано.

Обозначим через $V(x_1,x_2)$ функцию $\max(x_1,x_2)+1$ (*функция Вебба*).

Следствие 4. *При любом $k\geqslant 2$ функция $V(x_1,x_2)$ образует базис класса P_k.*

Доказательство. Имеем $V(x,x)=\bar{x}$. Далее суперпозициями функции $\bar{x}$ образуем функцию $x+k-1$. Теперь поолучаем $V(x_1,x_2)+k-1=\max(x_1,x_2)$. Следствие доказано.

Еще одно представление функций k-значной логики основано на функциях системы

$$0, 1, \ldots, k-1,\ x_1 + x_2 \pmod{k},\ x_1 \cdot x_2 \pmod{k},\ j_0(x), \ldots, j_{k-1}(x), \quad (1.6)$$

где

$$j_i(x) = \begin{cases} 1, & \text{если } x = i, \\ 0 & \text{в противном случае.} \end{cases}$$

Теорема 1.2. *Для любой функции $f(x_1, \ldots, x_n)$ из P_k имеет место представление*

$$f(x_1, \ldots, x_n) = \sum_{(\sigma_1, \ldots, \sigma_n) \in E_k^n} j_{\sigma_1}(x_1) \cdot \ldots \cdot j_{\sigma_n}(x_n) \cdot f(\sigma_1, \ldots, \sigma_n), \qquad (1.7)$$

где суммирование проводится по модулю k.

Доказательство легко получается непосредственной проверкой.

Следствие 1. *При любом $k \geqslant 2$ система функций* (1.6) *полна в классе P_k.*

Следствие 2. *При любом $k \geqslant 3$ система функций $\{j_0(x), x_1 + x_2\}$ полна в классе P_k.*

Доказательство. Константу 0 получаем в виде $x + \ldots + x$ (k раз), константу 1 — в виде $j_0(0)$. Затем из константы 1 и функции $x_1 + x_2$ получаем все остальные константы. Далее проверяем, что выполняются соотношения

$$j_i(x) = j_0(x + k - i) \quad \text{при} \quad 1 \leqslant i \leqslant k-1,$$

$$j_i(x_1) \cdot j_l(x_2) = j_2(j_i(x_1) + j_l(x_2)) \quad \text{при} \quad 0 \leqslant i, l \leqslant k-1.$$

Если $n \geqslant 3$ и $\sigma_1, \ldots, \sigma_n \in E_k$, то функцию $j_{\sigma_1}(x_1) \cdot \ldots \cdot j_{\sigma_n}(x_n)$ получаем последовательными подстановками функций

$$j_{\sigma_2}(x_2) \cdot j_1(y),\ \ldots,\ j_{\sigma_{n-2}}(x_{n-2}) \cdot j_1(y),\ j_{\sigma_{n-1}}(x_{n-1}) \cdot j_1(y),\ j_{\sigma_n}(x_n)$$

вместо переменной y в функции

$$j_{\sigma_1}(x_1) \cdot j_1(y), \quad j_{\sigma_1}(x_1) \cdot j_{\sigma_2}(x_2) \cdot j_1(y), \ldots$$
$$\ldots, j_{\sigma_1}(x_1) \cdot \ldots \cdot j_{\sigma_{n-2}}(x_{n-2}) \cdot j_1(y),\ j_{\sigma_1}(x_1) \cdot \ldots \cdot j_{\sigma_{n-1}}(x_{n-1}) \cdot j_1(y).$$

В заключение доказательства заметим, что сумму в правой части равенства (1.7) можно «собрать» из слагаемых вида $j_{\sigma_1}(x_1) \cdot \ldots \cdot j_{\sigma_n}(x_n)$ с помощью функции $x_1 + x_2$.

Теорема 1.3. *Пусть $k \geqslant 2$, R — коммутативное кольцо с единицей, определенное на множестве E_k. Тогда любую функцию из P_k можно представить полиномом над кольцом R в том и только том случае, когда R — поле.*

Доказательство. Пусть R — поле. Будем предполагать, что 0 — нейтральный элемент аддитивной группы поля R и 1 — нейтральный элемент мультипликативной группы поля R. Любой ненулевой элемент a поля R лежит в мультипликативной группе поля R, имеющей порядок $k-1$. Поэтому справедливо соотношение $a^{k-1} = 1$. Вместе с тем очевидно, что $0^{k-1} = 0$. Следовательно, для любого элемента x из E_k имеем

$$j_0(x) = 1 - x^{k-1},$$

где знак $-$ символизирует взятие обратного элемента в аддитивной группе поля R. Таким образом, функция j_0 реализуется полиномом над полем R. Это утверждение верно и для остальных функций j_i, поскольку

$$j_i(x) = j_0(x - i) \qquad (1 \leqslant i \leqslant k-1).$$

Используя теперь в представлении (1.7) в качестве сложения и умножения соответствующие операции поля R, получаем, что всякую функцию из P_k можно реализовать полиномом над полем R.

Предположим, что кольцо R полем не является. Покажем, что в этом случае в кольце R есть (ненулевые) делители нуля. Предположим, что это не так. Поскольку R не является полем, найдется такой ненулевой элемент a, что при умножении a на любой ненулевой элемент (например, справа) получится элемент, отличный от 1. Так как в качестве (правых) сомножителей рассматривается $k-1$ элемент, а в результате умножения получается не более $k-2$ элементов, то найдутся такие различные ненулевые элементы c_1, c_2, что $a \cdot c_1 = a \cdot c_2$. Иными словами, $a \cdot (c_1 - c_2) = 0$ и мы нашди два ненулевых делителя нуля.

Итак, пусть для некоторых ненулевых элементов b_1, b_2 поля R выполняется равенство $b_1 \cdot b_2 = 0$.

Предположим теперь, что функция j_0 реализуется полиномом над кольцом R:

$$j_0(x) = a_0 + a_1 x + \ldots + a_s x^s.$$

Из равенства $j_0(0) = 1$ следует, что $a_0 = 1$, а из равенств

$$b_1 \cdot b_2 = 0 = j_0(b_1) = 1 + a_1 b_1 + \ldots + a_s b_1^s$$

— что b_1 есть делитель числа 1. Иными словами, для некоторого элемента c из E_k имеем $b_1 \cdot c = 1$. Умножая обе части этого равенства на b_2, приходим к равенству $b_2 = 0$, что невозможно по предположению. Значит, функцию j_0 недьзя реализовать полиномом над кольцом R, и теорема доказана.

Следствие *Пусть R — кольцо вычетов по модулю k. Тогда система всех полиномов над кольцом R полна в классе P_k в том и только том случае, когда k — простое число.*

УПРАЖНЕНИЯ

2. Доказать, что при любом $k \geqslant 3$ система функций, полученная из системы (1.4) удалением функций $0, k-1, J_0, J_{k-1}$, полна в классе P_k.

3. Определим функцию $x \dot{-} y$:

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

Доказать, что при любом $k \geqslant 2$ система $\{\bar{x}, x \dot{-} y\}$ полна в классе P_k.

4. Доказать, что при любом $k \geqslant 2$ система функций $\{\bar{x}, \min(x, y)\}$ полна в классе P_k.

5. Доказать, что при любом $i \in E_k$ система функций $\{1, J_i(x), x + y \pmod k\}$ полна в P_k.

6. Доказать, что при любом $k \geqslant 2$ система функций $\{0, \ldots, k-1, J_0, \ldots, J_{k-1}, \max, x \cdot y\}$ полна в P_k.

7. Доказать, что при любом $k \geqslant 2$ система функций $P_k^{(1)} \cup \{x \cdot y\}$ полна в P_k.

§ 3. Алгоритм распознавания функциональной полноты

Сформулируем следующую *алгоритмическую проблему*: можно ли по произвольной конечной системе функций из P_k установить, является ли данная система функций полной в P_k? Иными словами, существует ли *алгоритм*, на вход которого поступает произвольная конечная система функций из P_k и который дает ответ «да» или «нет» в зависимости от полноты или неполноты рассматриваемой системы функций. Положительный ответ на этот вопрос был получен С.В. Яблонским [7, 2]. В теореме 1.4 излагается несколько обобщенная версия алгоритма С.В. Яблонского.

Теорема 1.4. *Существует алгоритм, который по произвольной конечной системе Q функций из P_k и функции f из P_k дает ответ на вопрос, принадлежит ли функция f множеству $[Q]$.*

Доказательство. Пусть заданы конечная система $Q = \{f_1, \ldots, f_s\}$ функций из P_k и функция $f \in P_k$. В целях упрощения рассуждений можно считать, что все функции $f_1, \ldots, f_s, f$ зависят от m переменных. По индукции определим монотонно неубывающую (по включению) последовательность

$$H_0, H_1, \ldots, H_r, \ldots \tag{1.8}$$

множеств функций от переменных $x_1, \ldots, x_m$.

Положим $H_0 = \varnothing$. Пусть уже определены множества $H_0, H_1, \ldots, H_r$. Для любого i $(1 \leqslant i \leqslant s)$ рассмотрим всевозможные формулы вида

$$f_i(h_1(x_1, x_2, \ldots, x_m), \ldots, h_m(x_1, x_2, \ldots, x_m)),$$

где функции $h_1, \ldots, h_m$ принадлежат множеству

$$H_r \cup \{e_1^m(x_1, x_2, \ldots, x_m), \ldots, e_m^m(x_1, x_2, \ldots, x_m)\}$$

(селекторные функции $e_1^m, \ldots, e_m^m$ здесь выбраны лишь для того, чтобы на каждом шаге построения использовать функции от m переменных; на самом деле вместо селекторных функций можно подставлять в функции f_i соответствующие переменные). Образуем множество H_{r+1}, добавляя к множеству H_r все функции, реализуемые указанными формулами.

Очевидно, что последовательность (1.8) монотонно не убывает. Кроме того, из построения непосредственно видно, что если $H_{r+1} = H_r$, то $H_{r+2} = H_{r+1}$ (если в конструкции множество H_r заменить равным ему множеством H_{r+1}, то получим иножество H_{r+2}, которое будет равно множеству H_{r+1}) и, следовательно, цепочка множеств стабилизируется:

$$H_r = H_{r+1} = H_{r+2} = \ldots$$

В каждом из множеств H_r не более k^{k^m} функций. Поэтому, просматривая множества цепочки (1.8) и сравнивая два соседних множества, мы не более чем через k^{k^m} шагов обнаружим наименьший номер r_0, начиная с которого наступает стабилизация. Очевидно, что процесс определения числа r_0 является полностью эффективным.

Итак, мы получили строго возрастающую цепочку множеств

$$H_0 \subset H_1 \subset \ldots \subset H_{r_0}.$$

Рассмотрим последнее множество H_{r_0} этой цепочки. Возможны два случая.

1. Функция f принадлежит множеству H_{r_0}.

Очевидно, что в этом случае $f \in [Q]$ (функции $e_1^m, \ldots, e_m^m$, используемые в построении, как отмечалось выше, можно заменить соответствующими переменными).

2. Функция f не принадлежит множеству H_{r_0}.

Здесь необходимо провести дополнительные рассуждения, чтобы показать, что $f \notin [Q]$. Именно, если бы выполнялось включение $f \in [Q]$, то функцию f можно было бы представить в виде

$$f(x_1, \ldots, x_m) = f_i(\Phi_1(x_1, \ldots, x_m), \ldots, \Phi_m(x_1, \ldots, x_m)), \tag{1.9}$$

где $1 \leqslant i \leqslant s$, а каждое выражение $\Phi_1, \ldots, \Phi_m$ есть либо формула над Q, либо символ переменной из числа $x_1, \ldots, x_m$.

Продолжая далее анализировать выражения $\Phi_1, \ldots, \Phi_m$, отличные от символов переменных, придем к «самым простым» подформулам из правой части формулы (1.9), которые имеют вид

$$f_j(x_{j_1}, \ldots, x_{j_m}), \tag{1.10}$$

где $j_1, \ldots, j_m \in \{1, 2, \ldots, m\}$. Формула (1.10), очевидно, реализует ту же функцию, что и формула

$$f_j(e_{j_1}^m(x_1, \ldots, x_m), \ldots, e_{j_m}^m(x_1, \ldots, x_m)).$$

Последняя функция по определению входит в множество H_1. Из формул вида (1.10) и символов переменных в формуле (1.9) составлены подформулы следующего уровня сложности:

$$f_l(\Psi_1(x_1, \ldots, x_m), \ldots, \Psi_m(x_1, \ldots, x_m)), \tag{1.11}$$

где $\Psi_1, \ldots, \Psi_m$ — либо формулы вида (1.10), либо символы переменных. Понятно, что формулы (1.11) реализуют функции из множества H_2 и т.д. В итоге приходим к выводу, что множество H_{r_0} состоит в точности из всех функций замыкания $[Q]$, которые зависят от переменных $x_1, \ldots, x_m$. Поэтому невхождение функции f в множество H_{r_0} равносильно ее невхождению в множество $[Q]$.

Итак, искомый алгоритм распознавания принадлежности функции f множеству $[Q]$ состоит в построении возрастающей цепочки множеств $H_0, H_1, \ldots$ до момента r_0 ее стабилизации и проверке выполнимости включения $f \in H_{r_0}$. Теорема доказана.

Следствие. *Существует алгоритм, который распознает полноту конечных систем функций в* P_k.

Доказательство. Достаточно, например, в качестве функции f рассмотреть функцию Вебба $V(x_1, x_2)$.

УПРАЖНЕНИЯ

8. Как может выглядеть алгоритм распознавания полноты конечных систем функций в P_k, которые заведомо содержат какую-либо из функций $\max,\ \min,\ x+y,\ x\cdot y,\ x\dot{-}y$?

§ 4. Теорема Кузнецова о функциональной полноте

Алгоритм распознавания полноты конечных систем функций, изложенный в предыдущем параграфе, требует большого объема вычислений и по этой причине на практике используется крайне редко. Альтернативой этому подходу служит подход, основанный на нахождении конечного числа «свойств», которые исчерпывающим образом характеризуют полные системы. Как обнаружилось, такими свойствами могут быть свойства принадлежности систем замкнутым класса определенного вида. Впервые этот подход был предложен А.В. Кузнецовым [7].

Прежде чем сформулировать и доказать теорему Кузнецова, введем необходимые определения и докажем две леммы. Пусть

$$G = \{g_1(y_1, \ldots, y_p), \ldots, g_r(y_1, \ldots, y_p)\}$$

и $g_1, \ldots, g_r \in P_k$. Говорят, что функция $f(x_1, \ldots, x_n)$ из P_k *сохраняет множество функций* G, если для любых $i_1, \ldots, i_n$ из $\{1, 2, \ldots, r\}$ выполняется соотношение

$$f(g_{i_1}(y_1, \ldots, y_p), \ldots, g_{i_n}(y_1, \ldots, y_p)) \in G.$$

Лемма 1.1. *Пусть* F *— множество всех функций из* P_k*, которые сохраняют множество функций* G*. Тогда* F *является замкнутым классом, содержащим все селекторные функции.*

Доказательство. Принадлежность селекторных функций множеству F легко следует из определений. Поэтому для доказательства замкнутости множества F достаточно убедиться в в том, что из принадлежности

функций $f_0, f_1, \ldots, f_m$ множеству F вытекает принадлежность множеству F функции h, где

$$h(x_1, \ldots, x_n) = f_0(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)).$$

Из включения $\{f_1, \ldots, f_m\} \subseteq F$ следует, что для любых $i_1, \ldots, i_n$ из $\{1, 2, \ldots, r\}$ функции

$$\begin{gathered} h_1(y_1, \ldots, y_p) = f_1(g_{i_1}(y_1, \ldots, y_p), \ldots, g_{i_n}(y_1, \ldots, y_p)), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ h_m(y_1, \ldots, y_p) = f_m(g_{i_1}(y_1, \ldots, y_p), \ldots, g_{i_n}(y_1, \ldots, y_p)) \end{gathered}$$

входят в множество G. Значит, в множество G будет входить функция

$$h_m(g_{i_1}(y_1, \ldots, y_p), \ldots, g_{i_n}(y_1, \ldots, y_p)) = f_0(h_1(y_1, \ldots, y_p), \ldots, h_m(y_1, \ldots, y_p)).$$

Лемма доказана.

Лемма 1.2. *Пусть множество G содержит функции $e_1^p, \ldots, e_p^p$, $G = [G]^{(p)}$, а F есть множество всех функций из P_k, сохраняющих множество функций G. Тогда $F^{(p)} = G$.*

Доказательство. Пусть $g_j, g_{i_1}, \ldots, g_{i_p} \in G$. Тогда в силу условия $[G]^{(p)} = G$ функция

$$g_j(g_{i_1}(y_1, \ldots, y_p), \ldots, g_{i_p}(y_1, \ldots, y_p))$$

принадлежит множеству G. Отсюда следует, что $g_j \in F$.

Обратно, если $f \in F^{(p)}$, то из условия $\{e_1^p, \ldots, e_p^p\} \subseteq G$ и равенства

$$f(e_1^p(y_1, \ldots, y_p), \ldots, e_p^p(y_1, \ldots, y_p)) = f(y_1, \ldots, y_p)$$

следует, что $f \in G$. Лемма доказана.

Замкнутый класс $R \subseteq P_k$ называется *предполным* в P_k, если $R \neq P_k$ и $[R \cup \{f\}] = P_k$ для любой функции f из множества $P_k \setminus R$.

Теорема 1.5. *Для любого $k \geqslant 2$ число предполных в P_k классов конечно; произвольная система функций из P_k полна в P_k тогда и только тогда, когда она целиком не содержится ни в одном из предполных классов.*

Доказательство. Образуем систему $\{G_1, \ldots, G_m\}$ всех собственных подмножеств множества $P_k^{(2)}$, которые обладают следующими двумя свойствами.

1. Каждое множество G_i содержит функции e_1^2, e_2^2.
2. $[G_i]^{(2)} = G_i \quad (i = 1, 2, \ldots, m)$.

Пусть F_i' — множество всех функций из P_k, которые сохраняют множество функций G_i. В силу лемм 1.1, 1.2 множество F_i'является замкнутым классом и выполняется соотношение $(F_i')^{(2)} = G_i$. Выделим в семействе $\{F_1', \ldots, F_m'\}$ все максимальные по включению классы. Обозначим их $F_1, \ldots, F_l$. Положим $\mathcal{F} = \{F_1, \ldots, F_l\}$. Покажем, что $\mathcal{F}$ есть искомое семейство предполных классов.

Прежде всего, из построения следует, что каждый класс F_i отличен от P_k. Пусть S — произвольное множество функций из P_k. Если S целиком содержится в некотором классе семейства $\mathcal{F}$, то неполнота S следует из замкнутости классов семейства $\mathcal{F}$ и несовпадении их с P_k.

Предположим, что множество S целиком не входит ни в один из классов семейства $\mathcal{F}$. Тогда, конечно, и замыкание S не будет целиком входить ни в один из классов семейства $\mathcal{F}$. Поскольку нас интересует полнота (неполнота) множества S, можно считать, что S — замкнутый класс. Нетрудно видеть, что множество $S \cup \{e_1^2, e_2^2\}$ полно или неполно в классе P_k одновременно с полнотой или неполнотой класса S. Поэтому далее можно предполагать, что класс S содержит функции e_1^2, e_2^2.

Положим $G = S^{(2)}$. Тогда множество G содержит функции e_1^2, e_2^2 и удовлетворяет условию $[G]^{(2)} = G$ (поскольку замкнутый класс S, очевидно, сохраняет множество функций G). Предполагая, что $G \neq P_k^{(2)}$, обозначим через множество F_i' всех функций, сохраняющих G. Тогда, конечно, выполняется включение $S \subseteq F_i'$. Однако класс F_i' целиком содержится в некотором максимальном классе F_j. Значит, приходим к включению $S \subseteq F_j$, что невозможно по предположению. Следовательно, класс S совпадает с P_k.

Остается показать, что каждый из классов $F_1, \ldots, F_l$ предполон в P_k. Однако если, например, $f \notin F_i$, то система функций $F_i \cup \{f\}$ не содержится целиком ни в одном из классов семейства $\mathcal{F}$ (используем свойство максимальности классов семейства $\mathcal{F}$). Следовательно, по доказанному система $F_i \cup \{f\}$ полна в классе P_k.

УПРАЖНЕНИЯ

9. Выписать все детали доказательства теоремы 1.5 для случая $k = 2$.

§ 5. Замкнутые классы, не имеющие конечных базисов

Как установил Э. Пост [10, 11] (см. также [6]), всякий замкнутый класс булевых функций имеет конечный базис. Однако при любом $k \geqslant 3$ в P_k существуют закнутые классы, которые не имеют базисов вообще, а также замкнутые классы со счетными базисами. Эти результаты получены Ю.И. Яновым и А.А. Мучником [9].

Теорема 1.6 (Ю.И. Янов). *Для любого $k \geqslant 3$ в P_k существует замкнутый класс, не имеющий базиса.*

Доказательство. Определим в P_k последовательность функций f_0, $f_1, \ldots$: пусть $f_0(x_1) = 0$ и при $n \geqslant 1$

$$f_n(x_1, \ldots, x_n) = \begin{cases} 1, & \text{если } x_1 = \ldots = x_n = 2, \\ 0 & \text{в остальных случаях.} \end{cases}$$

Положим $F = [\{f_0, f_1, \ldots\}]$. Отметим два свойства функций f_n.

1. Если $n \geqslant 1$ и набор $(i_1, \ldots, i_n)$ содержит m различных значений, то формула $f_n(x_{i_1}, \ldots, x_{i_n})$ реализует функцию f_m.

2. Любая суперпозиция вида $f_n(\ldots, f_m(\ldots), \ldots)$ реализует функцию, тождественно равную нулю (поскольку функция f_m не принимает значения 2).

Из этих свойств следует, что класс F помимо функций $f_1, f_2, \ldots$ содержит только тождественно нулевые функции от любого числа переменных.

Докажем, что класс F не имеет базиса. Пусть, напротив, класс F имеет базис B. Тогда по свойству 1 в базис B не могут входить две функции f_m, f_n, где $1 \leqslant m < n$. Если же базис B содержит только одну функцию f_m, где $m \neq 0$, то в силу свойства 2 из нее (и, возможно, функции f_0, если $f_0 \in B$) нельзя получить функцию f_n, где $n > m$. Противоречие показывает, что класс F не имеет базиса. Теорема доказана.

Теорема 1.7 (А.А. Мучник). *Для любого $k \geqslant 3$ в P_k существует замкнутый класс, имеющий счетный базис.*

Доказательство. Определим в P_k последовательность функций g_2, $g_3, \ldots$:

$$g_n(x_1, \ldots, x_n) = \begin{cases} 1, & \text{если } x_1 = \ldots = x_{i-1} = x_{i+1} = \ldots = x_n = 2, \\ & x_i = 1 \quad (1 \leqslant i \leqslant n), \\ 0 & \text{в остальных случаях.} \end{cases}$$

Положим $G = [\{g_2, g_3, \ldots\}]$. Докажем, что система функций $\{g_2, g_3, \ldots\}$ образует базис класса G. Для этого достаточно установить, что никакая функция g_m не выражается формульно через остальные функции системы $\{g_2, g_3, \ldots\}$. Предположим, что это не так, т.е.

$$g_m(x_1, \ldots, x_m) = g_s(\Phi_1, \ldots, \Phi_s), \tag{1.12}$$

где $s \neq m$ и $\Phi_1, \ldots, \Phi_s$ — либо формулы над множеством функций $\{g_2, \ldots, g_{m-1}, g_{m+1}, \ldots\}$, либо символы переменных. Возможны три случая.

1. Среди выражений $\Phi_1, \ldots, \Phi_s$ имеются по крайней мере две формулы.

Пусть это будут Φ_i и Φ_j. Обращаясь к определению функций g_n, видим, что функции, реализуемые формулами Φ_i, Φ_j, не принимают значения 2. Следовательно, функция, реализуемая формулой из правой части равенства (1.12), не может принимать значение 1. Это противоречит определению функции g_m.

2. Среди выражений $\Phi_1, \ldots, \Phi_s$ только одно отлично от символа переменной.

Пусть это будет Φ_i. Так как $s \geqslant 2$, то имеется хотя бы одно выражение Φ_j, которое совпадает с символом переменной. Пусть, например, это будет переменная x_l. Рассмотрим следующие значения переменных:

$$x_1 = \ldots = x_{l-1} = x_{l+1} = \ldots = x_m = 2, \quad x_l = 1. \tag{1.13}$$

Для этих значений переменных функция, реализуемая формулой Φ_i, примет значение 0 или 1, а функция, реализуемая формулой $g_s(\Phi_1, \ldots, \Phi_s)$, — значение 0. Вновь получаем противоречие.

3. Все выражения $\Phi_1, \ldots, \Phi_s$ суть символы переменных.

Очевидно, что в этом случае должно быть $s > m$. Значит, среди выражений $\Phi_1, \ldots, \Phi_s$ по крайней мере два раза встретится одна и та же переменная x_l $(1 \leqslant l \leqslant m)$. Рассмотрев значения переменных (1.13), придем к выводу, что функция, реализуемая формулой $g_s(\Phi_1, \ldots, \Phi_s)$, принимает на этом наборе значение 0. Следовательно, и этот случай невозможен. Теорема доказана.

Следствие. *Для любого $k \geqslant 3$ семейство всех замкнутых классов в P_k имеет континуальную мощность.*

Доказательство. В самом деле, если $\{m_1, m_2, \ldots\}$ и $\{n_1, n_2, \ldots\}$ — два различных непустых подмножества множества $\{2, 3, \ldots\}$, то, как

установлено в теореме 1.7, множества

$$\{g_{m_1}, g_{m_2}, \ldots\}, \quad \{g_{n_1}, g_{n_2}, \ldots\}$$

будут являться базисами различных замкнутых классов. Таким образом, в P_k существует по меньшей мере континуальное семейство замкнутых классов с конечным либо счетным базисом. С другой стороны, мощность множества замкнутых классов в P_k не превосходит мощности множества всех подмножеств счетного множества P_k, т.е. не превосходит мощности континуума. Следствие доказано.

УПРАЖНЕНИЯ

10. Доказать, что замкнутый класс F из теоремы 1.6 не имет предполных классов.

11. Доказать, что при любом $k \geqslant 4$ в P_k существует континуальное число замкнутых классов, не имеющих базиса.

Глава 2

КОНЕЧНЫЕ АВТОМАТЫ-РАСПОЗНАВАТЕЛИ

§ 1. Конечный автомат без выхода. Конечно-автоматные множества

Будем рассматривать алфавит A, состоящий из элементов $a_1, \ldots, a_k$, которые называем буквами алфавита A. Конечную последовательность букв алфавита A, записанных без пропусков одна за другой, считаем словом в алфавите A. Множество всех слов в алфавите A обозначаем через A^*. В множество A^* включаем пустое слово Λ — слово, не содержащее букв. На множестве A^* введем операцию *конкатенации* (соединения) слов: если $\bar{a}, \bar{b}$ — слова в алфавите A^*, то конкатенацией слов $\bar{a}, \bar{b}$ называется слово $\bar{a}\bar{b}$, полученное из слова $\bar{a}$ приписыванием справа слова $\bar{b}$. Отметим, что для произвольного слова $\bar{a}$ выполняются равенства $\Lambda\bar{a} = \bar{a}\Lambda = \bar{a}$, в частности, $\Lambda\Lambda = \Lambda$. Для любого слова $\bar{a}$ и любого натурального числа n через $\bar{a}^n$ будем обозначать слово, полученное конкатенацией n экземпляров слова $\bar{a}$. Полагаем также $\bar{a}^0 = \Lambda$.

Прежде чем дать определение конечного автомата без выхода, мы хотим сформулировать несколько содержательных предпосылок, которые помогут уяснить суть вводимых понятий.

Во-первых, мы хотим, чтобы автомат мог воспринимать (читать) любые слова в алфавите A — входном алфавите автомата. Обычно говорят, что слово $\bar{a} = a_{i_1}a_{i_2}\ldots a_{i_n}$ (в алфавите A) подается на вход автомата (или читается автоматом). Поскольку слово $\bar{a}$ может иметь произвольную длину, автомат воспринимает слово $\bar{a}$ не все сразу, а побуквенно: сначала букву a_{i_1}, затем a_{i_2} и т.д. В связи с этим считают, что процесс чтения слова $\bar{a}$ автоматом происходит в дискретные моменты времени $t = 1, 2, \ldots$. Кроме того, в эти моменты времени в автомате должны происходить изменения «состояния». Именно, считается, что в каждый момент времени автомат может находиться в одном из конечного числа состояний. При подаче на вход автомата очередной буквы входного слова автомат, как говорят, переходит из одного состояния в другое состояние (оно может совпадать с предыдущим состоянием). Таким образом, имеется некоторая управляющая функция автомата — *функция переходов*,

которая по букве алфавита A и «текущему» состоянию автомата определяет состояние автомата в следующий момент времени. Изложенные соображения мы формализуем в виде понятия *простейшего автомата.*

Итак, *простейший автомат* $\mathcal{A}$ задается *входным алфавитом* $A = \{a_1, \ldots, a_k\}$, конечным множеством *состояний* $Q = \{q_1, \ldots, q_r\}$ и *функцией переходов* f, которая отображает множество $A \times Q$ в множество Q. Символически это будем изображать так: $\mathcal{A} = (A, Q, f)$. Учитывая приведенное выше содержательное описание работы автомата $\mathcal{A}$, обозначим через $x(t)$ букву алфавита A, подаваемую на вход автомата $\mathcal{A}$ в момент времени t, а через $q(t)$ — его состояние в этот момент времени. Тогда функционирование (во времени) автомата $\mathcal{A}$ можно выразить уравнением

$$q(t) = f(x(t), q(t-1)). \tag{2.1}$$

Из этого уравнения, равно как из описания работы автомата, видно, что перед подачей слова $\bar{a}$ автомат $\mathcal{A}$ необходимо как-то «настроить», привести в исходное состояние. Иными словами, мы должны решить, в каком из состояний множества Q автомат $\mathcal{A}$ начнет воспринимать первую букву слова $\bar{a}$ — какое значение мы должны выбрать в уравнении (2.1) для величины $q(0)$. Это состояние автомата $\mathcal{A}$ обычно называют *начальным* состоянием, а автомат с выделенным начальным состоянием — *инициальным* автоматом. В качестве начального состояния мы, если не оговаривается иное, будем выбирать состояние q_1. Теперь инициальный автомат $\mathcal{A}$ будет определяться четверкой (A, Q, f, q_1). В свою очередь, к уравнению (2.1) мы добавим начальное условие

$$q(0) = q_1. \tag{2.2}$$

Соотношения (2.1),(2.2) будем называть *каноническими уравнениями* автомата $\mathcal{A}$.

Существует еще несколько способов задания инициального автомата. Один из наиболее распространенных — задание с помощью *диаграмм Мура.* Диаграмма Мура представляет собой графическое задание автомата. Чтобы построить диаграмму Мура автомата $\mathcal{A} = (A, Q, f, q_1)$ с r состояниями, на плоскости выбирают r различных точек (или непересекающихся кругов), которые обозначают символами $q_1, \ldots, q_r$. Из каждой точки диаграммы Мура проводят ровно k направленных дуг, которые ведут в точки этой диаграммы. Каждая дуга помечается одной буквой алфавита A (разные дуги, выходящие из одной точки, — разными буквами). При этом дуга, выходящая из точки q_j и помеченная буквой a_i, ведет

в точку q_l, где $q_l = f(a_i, q_j)$. Начальное состояние q_1 обычно помечается символом $*$. На рис. 1 изображены примеры диаграмм Мура.

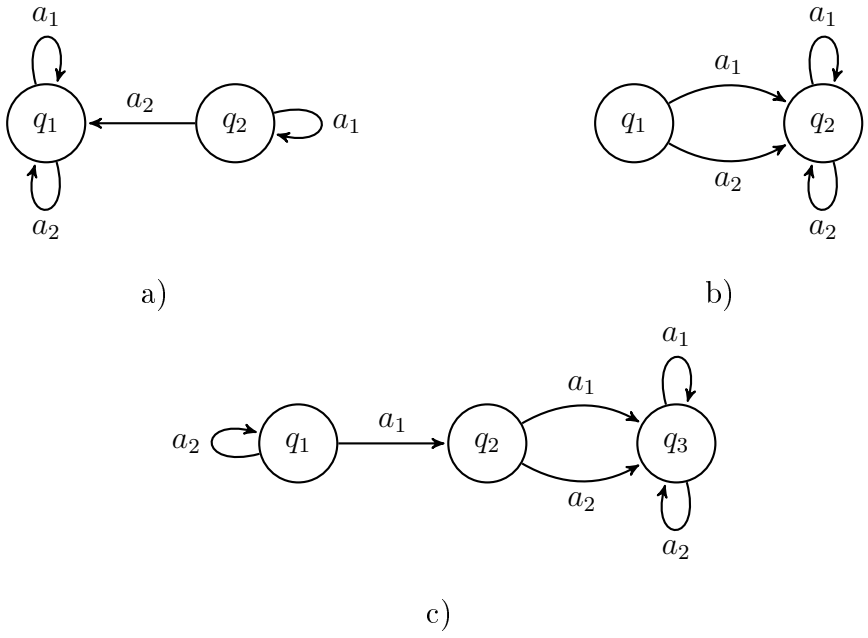


Рис. 1

Под действием входного слова $\bar{a} = a_{i_1} a_{i_2} \dots a_{i_n}$ автомат $\mathcal{A}$ последовательно переходит из одного состояния в другое. Перед поступлением буквы a_{i_1} автомат $\mathcal{A}$ находится в начальном состоянии q_1. Под действием буквы a_{i_1} автомат $\mathcal{A}$ переходит из состояния q_1 в состояние $q_{j_2} = f(a_{i_1}, q_1)$, далее под действием буквы a_{i_2} — из состояния q_{j_2} в состояние $q_{j_3} = f(a_{i_2}, q_{j_2})$ и т.д. Условимся считать, что под действием пустого слова Λ автомат $\mathcal{A}$ никуда не переходит, т.е. остается в начальном состоянии q_1.

Определенный нами простейший инициальный автомат пока обладает одним существенным недостатком: несмотря на то, что на вход автомата мы можем подавать произвольные слова в алфавите A, никаких сигналов о функционировании автомата мы в ответ получать не можем. Мы бы хотели, как минимум, чтобы автомат после прочтения входного слова выдавал сигналы типа «да» и «нет». Иными словами, мы хотим, чтобы некоторые слова в алфавите A автомат «допускал» (принимал, распознавал), другие слова — «отвергал».

Этой цели можно достичь различными способами. Самый простой из них — выделить в множестве Q некоторое подмножество F «заключительных» (финальных) состояний и считать, что слово $\bar{a}$ *допускается*

(принимается, распознается) автоматом $\mathcal{A}$, если после подачи слова $\bar{a}$ автомат $\mathcal{A}$ оказывается в одном из состояний множества F. В противном случае считаем, что слово $\bar{a}$ *отвергается* автоматом $\mathcal{A}$. Таким образом, попадание в одно из состояний множества F — это и есть сигнал «да», который «выдает» автомат в ответ на поступившее на его вход слово.

Теперь мы можем дать вполне строгое определение *автомата без выхода*. Итак, *автомат без выхода* — это математическая модель распознающего устройства с конечной памятью, которая задается набором (A, Q, f, q_1, F), где $A = \{a_1, \ldots, a_k\}$ — входной алфавит автомата, $Q = \{q_1, \ldots, q_r\}$ — множество состояний автомата, f — функция переходов автомата, отображающая множество $A \times Q$ в множество Q, q_1 — начальное состояние автомата и F — непустое множество заключительных состояний автомата. Отметим, что начальное состояние q_1 может быть одновременно и заключительным состоянием.

В этой главе автомат без выхода будем называть просто автоматом. С автоматом $\mathcal{A}$, как это объяснено выше, связываем множество $D(\mathcal{A})$ всех слов в алфавите A, которые допускаются автоматом $\mathcal{A}$. Далее такие множества $D(\mathcal{A})$ будем называть *конечно-автоматными* множествами. В параграфах 2–4 мы исследуем конечно-автоматные множества с различных точек зрения.

Начнем с некоторых простых примеров. Рассмотрим автоматы, диаграммы Мура которых изображены на рис. 1. Если в случае a) положить $F = \{q_1\}$, то соответствующий автомат будет допускать множество A^*, если же взять $F = \{q_2\}$ — то пустое множество $\varnothing$ (поскольку состояние q_2 не достижимо из состояния q_1). Более интересным представляется случай b): при $F = \{q_1\}$ автомат допускает множество $\{\Lambda\}$, состоящее только из пустого слова, при $F = \{q_2\}$ — множество A^* без пустого слова. В случае c) выбор $F = \{q_1\}$, $F = \{q_2\}$ или $F = \{q_3\}$ дает соответственно: множество всех слов в алфавите $\{a_2\}$ (включая пустое слово); множество всех слов вида $a_2^n a_1$, где $n \geqslant 0$; множество всех слов вида $a_2^n a_1 \bar{a}$, где $n \geqslant 0$ и $\bar{a} \neq \Lambda$.

Рассмотрим еще два примера несколько иного характера. Как видно из рис. 2, входной алфавит автоматов состоит из одной буквы a_1. Нетрудно видеть, что в случае a) в зависимости от выбора $F = \{q_1\}$ или $F = \{q_2\}$ автомат будет допускать либо все слова, имеющие четную длину (включая пустое слово Λ, которое имеет длину 0), либо все слова, имеющие нечетную длину. Похожая картина наблюдается и в случае b): если $F = \{q_1\}$, $F = \{q_2\}$ или $F = \{q_3\}$, то автомат допускает все слова, имею-

щие соответственно длину вида $3n$, $3n+1$ или $3n+2$ ($n \geqslant 0$). Эти примеры легко обобщить на множества слов с длинами вида $kn + l$ ($n \geqslant 0$), где $k \geqslant 4$ и $0 \leqslant l < k$.

a) b)

Рис. 2

УПРАЖНЕНИЯ

1. Построить диаграмму Мура инициального автомата $\mathcal{A} = (A, Q, f, q_1)$, где $A = \{a_1, a_2\}$, $Q = \{q_1, q_2, q_3, q_4\}$, а функция переходов f задается таблицей

	q_1	q_2	q_3	q_4
a_1	q_2	q_1	q_4	q_4
a_2	q_3	q_4	q_1	q_2

2. Построить диаграмму Мура для автоматов с входным алфавитом $\{a_1, a_2\}$, которые допускают следующие множества:
a) $\{a_1, a_2a_1\}$; b) все слова длины 3; c) все слова, которые начинаются словом a_1a_2; d) все слова, которые оканчиваются словом a_1a_1; e) все слова, которые содержат слово $a_2a_2a_1$ (в частности, начинаются и оканчиваются этим словом).

§ 2. Правоинвариантная эквивалентность. Теоретико-множественные операции над конечно-автоматными множествами

Пусть $\mathcal{A} = (A, Q, f, q_1)$ — инициальный автомат, $Q = \{q_1, \ldots, q_r\}$. Будем предполагать, что каждое состояние из Q *достижимо* из начального состояния q_1, т.е. для каждого состояния q_j из Q найдется такое слово в алфавите A, под действием которого автомат $\mathcal{A}$ переходит из состояния q_1 в состояние q_j.

Разобьем множество A^* всех слов в алфавите A на r попарно не пересекающихся подмножеств (классов), относя к классу K_i те и только те

слова, которые переводят автомат $\mathcal{A}$ из состояния q_1 в состояние q_i. Разбиение $\{K_1, \ldots, K_r\}$ множества A^* на классы порождает на множестве A^* отношение эквивалентности $\sim$: для любых двух слов $\bar{a}, \bar{b} \in A^*$ полагаем $\bar{a} \sim \bar{b}$ в том и только том случае, когда слова $\bar{a}, \bar{b}$ входят в один и тот же класс K_i. Отношение $\sim$ действительно представляет собой отношение эквивалентности: оно рефлексивно (для любого слова $\bar{a}$ отношение $\bar{a} \sim \bar{a}$ выполняется тривиальным образом), оно симметрично (из $\bar{a} \sim \bar{b}$ следует $\bar{b} \sim \bar{a}$) и оно транзитивно (из $\bar{a} \sim \bar{b}$ и $\bar{b} \sim \bar{c}$ следует $\bar{a} \sim \bar{c}$). Будем называть отношение $\sim$ *конечно-автоматной эквиваленостью*, порожденной автоматом $\mathcal{A}$.

Помимо определяющих свойств отношения эквивалентности отношение $\sim$ обладает еще одним свойством, характерным только для автоматов. Это свойство *правой инвариантности*: если $\bar{a} \sim \bar{b}$ и $\bar{c}$ — любое слово в алфавите A, то $\bar{a}\bar{c} \sim \bar{b}\bar{c}$. Свойство правой инвариантности для отношения $\sim$ легко устанавливается непосредственно из определения отношения $\sim$ и разбиения $\{K_1, \ldots, K_r\}$: если под действием слов $\bar{a}, \bar{b}$ автомат $\mathcal{A}$ переходит из состояния q_1 в одно и то же состояние из множества Q, то, разумеется, под действием слов $\bar{a}\bar{c}$ и $\bar{b}\bar{c}$ автомат $\mathcal{A}$ также будет переходить из состояния q_1 в одно и то же состояние из Q.

Рассмотрим на множестве A^* произвольное правоинвариантное отношение эквивалентности $\sim$. Оно задает разбиение множества A^* на попарно не пересекающиеся классы: к одному классу относятся все слова, которые эквивалентны в смысле отношения $\sim$. Вообще говоря, число классов отношения $\sim$ может оказаться бесконечным. Будем говорить, что отношение $\sim$ имеет *конечный индекс*, если число таких классов конечно.

Пусть на множестве A^* задано провоинвариантное отношение эквивалентности $\sim$ конечного индекса и $K_1, \ldots, K_r$ — все классы этой эквивалентности. Будем предполагать, что слово Λ входит в класс K_1. Определим конечный инициальный автомат $\mathcal{K}$ с входным алфавитом A и множеством состояний $\{K_1, \ldots, K_r\}$. Начальным состоянием автомата $\mathcal{K}$ объявим класс K_1, а функцию переходов h определим следующим образом: если $a_i \in A$, $\bar{b}$ — слово из класса K_j, то $h(a_i, K_j)$ есть тот (единственный) класс K_l, который содержит слово $\bar{b}a_i$. Корректность определения функции h следует из свойства правой инвариантности отношения $\sim$: если $\bar{b}, \bar{c}$ — любые слова из класса K_j, то $\bar{b}a_i, \bar{c}a_i$ будут принадлежать одному и тому же классу K_l.

Легко видеть, что все состояния автомата $\mathcal{K}$ достижимы из начально-

го состояния K_1: если $\bar{b}$ — слово из класса K_j, то, «применяя» к классу K_1 слово $\bar{b}$, получим класс K_j, поскольку $\Lambda \in K_1$ и $\bar{b}\Lambda = \bar{b}$. Кроме того, непосредственно из определения автомата $\mathcal{K}$ видно, что конечно-автоматная эквивалентность, порожденная автоматом $\mathcal{K}$, совпадает с эквивалентностью $\sim$. Таким образом, мы убедились в справедливости следующего утверждения.

Теорема 2.1. *Каждый конечный инициальный автомат порождает правоинвариантное отношение эквивалентности конечного индекса. Обратно, каждое правоинвариантное отношение эквивалентности конечного индекса есть конечно-автоматное отношение эквивалентности.*

Применим полученный результат, чтобы охарактеризовать конечно-автоматные множества. Пусть $\mathcal{A} = (A, Q, f, q_1, F)$ — конечный автомат, $F = \{q_{j_1}, \ldots, q_{j_s}\}$ и $X = D(\mathcal{A})$. Согласно определениям из § 1, конечно-автоматное множество X состоит из всех слов в алфавите A, которые переводят автомат $\mathcal{A}$ из начального состояния q_1 в одно из состояний $q_{j_1}, \ldots, q_{j_s}$. Будем предполагать, что все состояния $q_{j_1}, \ldots, q_{j_s}$ достижимы из состояния q_1. Понятно, что в этом случае множество X можно представить в виде объединения s попарно не пересекающихся множеств $X_1, \ldots, X_s$, где X_t состоит из всех слов в алфавите A, которые переводят автомат $\mathcal{A}$ из состояния q_1 в состояние q_{j_t} $(1 \leqslant t \leqslant s)$. Однако, как мы определили выше, множество X_t есть класс эквивалентных слов в конечно-автоматной эквивалентности, порожденной автоматом $\mathcal{A}$. Учитывая теперь теорему 2.1, приходим к следующему результату.

Теорема 2.2. *Всякое непустое конечно-автоматное множество есть объединение некоторого числа классов подходящего правоинвариантного отношения эквивалентности конечного индекса. Обратно, объединение любого числа классов произвольного провоинвариантного отношения эквивалентности конечного индекса есть конечно-автоматное множество.*

Теорема 2.2 дает нам первое описание конечно-автоматных множеств, не связанное с понятием конечного автомата. Его удобно использовать для доказательства непринадлежности некоторых множеств классу конечно-автоматных множеств. Для примера рассмотрим в алфавите $\{a_1, a_2\}$ множество X, состоящее из всех слов вида $a_1^n a_2^n$ $(n \geqslant 1)$. Докажем, что оно не является конечно-автоматным. Пусть это не так. Тогда для подходящего правоинвариантного отношения эквивалентности

$\sim$ конечного индекса множество X есть объединение некоторого числа классов этой эквивалентности. Поскольку эквивалентность $\sim$ имеет лишь конечное число классов, найдутся такие неравные числа m, n, что слова a_1^m, a_1^n принадлежат одному и тому же классу эквивалентности $\sim$, т.е. $a_1^m \sim a_1^n$. Ввиду свойства правой инвариантности отношения $\sim$ получаем далее $a_1^m a_2^n \sim a_1^n a_2^n$. Однако слово $a_1^n a_2^n$ принадлежит множеству X. Значит, слово $a_1^m a_2^n$ также принадлежит множеству X. Это противоречит определению множества X.

Чтобы получать примеры более сложных конечно-автоматных множеств, рассмотрим ряд операций, которые сохраняют конечную автоматность множеств. Первыми в этом ряду будут теоретико-множественные операции дополнения, объединения и пересечения.

Пусть задан автомат $\mathcal{A} = (A, Q, f, q_1, F)$ и $X = D(\mathcal{A})$. Понятно, что все множество A^* разбивается на два непересекающихся множества: множество всех слов, под действием которых автомат $\mathcal{A}$ из состояния q_1 попадает в одно из состояний множества F (по определению это есть множество X), и множество всех остальных слов, под действием которых автомат $\mathcal{A}$ из состояния q_1 попадает в состояния из $Q \setminus F$. Следовательно, множество $\bar{X} = A^* \setminus X$ допускается автоматом $\mathcal{A}_1 = (A, Q, f, q_1, Q \setminus F)$. Как видно, автомат $\mathcal{A}_1$ отличается от автомата $\mathcal{A}$ только выбором множества заключительных состояний.

Итак, операция дополнения (до множества A^*) не выводит за пределы класса конечно-автоматных множеств. Этот результат мы могли бы доказать и по-другому, воспользовавшись теоремой 2.2. В самом деле, согласно теореме 2.2 для подходящего правоинвариантного отношения эквивалентности $\sim$ конечного индекса множество X есть объединение некоторого числа классов эквивалентности $\sim$. Понятно, что множество $\bar{X}$ при этом будет являться объединением оставшихся классов эквивалентности $\sim$. Значит, множество $\bar{X}$ также конечно-автоматно.

Перейдем к операциям объединения и пересечения. Мы хотим показать, что эти операции также не выводят за пределы класса конечно-автоматных множеств. Ввиду законов де Моргана достаточно рассмотреть только одну из этих операций, например, операцию пересечения.

Пусть X, Y — конечно-автоматные множества, а правоинвариантные отношения эквивалентности $\sim_1$ и $\sim_2$ конечного индекса выбраны так, что X есть объединение некоторого числа классов отношения $\sim_1$ и Y есть объединение некоторого числа классов отношения $\sim_2$. Пусть далее $K_1, \ldots, K_s$ — все классы отношения $\sim_1$, $L_1, \ldots, L_t$ — все классы отноше-

ния $\sim_2$ и, например,

$$X = K_1 \cup \ldots \cup K_u, \quad Y = L_1 \cup \ldots \cup L_v.$$

Введем на множестве A^* новое отношение эквивалентности $\sim_3$ (пересечение отношений $\sim_1$ и $\sim_2$). Для этого, отправляясь от разбиений $\{K_1, \ldots, K_s\}$ и $\{L_1, \ldots, L_t\}$ множества A^*, создадим новое разбиение $\{M_1, \ldots, M_p\}$ множества A^*: множества $M_1, \ldots, M_p$ — это все непустые пересечения вида $K_i \cap L_j$, где $1 \leqslant i \leqslant s$ и $1 \leqslant j \leqslant t$. Понятно, что $p \leqslant st$ и $\{M_1, \ldots, M_p\}$ — действительно разбиение множества A^* на непустые попарно не пересекающиеся множества. Значит, на основе разбиения $\{M_1, \ldots, M_p\}$ можно определить отношение эквивалентности $\sim_3$, полагая слова $\bar{a}, \bar{b} \in A^*$ эквивалентными в смысле отношения $\sim_3$ в том и только том случае, когда $\bar{a}, \bar{b}$ принадлежат одному и тому же множеству разбиения $\{M_1, \ldots, M_p\}$.

Легко видеть, что отношение эквивалентности $\sim_3$ провоинвариантно: если $\bar{a} \sim_3 \bar{b}$, то по определению эквивалентности $\sim_3$ будет также $\bar{a} \sim_1 \bar{b}$ и $\bar{a} \sim_2 \bar{b}$. Из последних двух соотношений следует, что для любого слова $\bar{c}$ имеем $\bar{a}\bar{c} \sim_1 \bar{b}\bar{c}$ и $\bar{a}\bar{c} \sim_2 \bar{b}\bar{c}$. Вспоминая, что $\sim_3$ есть пересечение отношений эквивалентности $\sim_1$ и $\sim_2$, получаем, что $\bar{a}\bar{c} \sim_3 \bar{b}\bar{c}$. Кроме того, из определения следует, что $\sim_3$ есть отношение конечного индекса. Остается представить множество $X \cap Y$ в виде объединения некоторого числа множеств $M_1, \ldots, M_p$. Понятно, что это будут в точности все множества M_l, которые являются непустыми пересечениями множеств из числа $K_1, \ldots, K_u$ с множествами из числа $L_1, \ldots, L_v$.

Полученный выше результат мы оформим в виде теоремы.

Теорема 2.3. *Операции дополнения, объединения и пересечения сохраняют конечную автоматность множеств.*

С использованием теоремы 2.3 можно получить аналогичные результаты и для других теоретико-множественных операций, например, для операций разности и симметрической разности:

$$X \setminus Y = X \cap \bar{Y}, \quad X \triangle Y = (X \setminus Y) \cup (Y \setminus X).$$

УПРАЖНЕНИЯ

3. Построив на множестве $\{a_1, a_2\}^*$ подходящие правоинвариантные отношения эквивалентности, доказать конечную автоматность следующих множеств:

$$a)\ \{\Lambda\}; \quad b)\ \{a_1\}; \quad c)\ \{\Lambda, a_1, a_2\}; \quad d)\ \{a_1^n a_2 :\ n \geqslant 0\}.$$

4. Для любого $n \geqslant 2$ определить на множестве $\{a_1, a_2\}^*$ правоинвариантную эквивалентность, имеющую ровно n классов эквивалентности.

5. По аналогии с правоинвариантной эквивалентностью определить на множестве A^* левоинвариантную эквивалентность: если $\bar{a} \sim \bar{b}$ и $\bar{c}$ — произвольное слово из A^*, то $\bar{c}\bar{a} \sim \bar{c}\bar{b}$. Доказать для левоинвариантной эквивалентности аналог теоремы 2.2.

6. Пользуясь теоремой 2.3, доказать конечную автоматность следующих множеств:
a) любое конечное множество слов в алфавите A; b) любое множество вида $A^* \setminus X$, где X — конечное множество слов в алфавите A; c) множество всех слов в алфавите A, имеющих вид a_i^n где $1 \leqslant i \leqslant k$ и $n \geqslant 1$; d) множество всех слов в алфавите $\{a_1, a_2\}$, которые имеют четную длину, начинаются буквой a_1 и в которых буквы a_1, a_2 чередуются.

§ 3. Недетерминированные автоматы

Далее мы рассмотрим две новые, сугубо автоматные операции. Чтобы показать, что они не выводят за пределы класса конечно-автоматных множеств, нам придется прибегнуть к техническому приему, который проще всего объяснить, если несколько расширить введенное выше понятие автомата. Речь пойдет о так называемых *недетерминированных* автоматах. С содержательной точки зрения недетерминированный автомат отличается от ранее рассмотренного (детерминированного) автомата только тем, что функция перехода недетерминированного автомата позволяет ему в каждом состоянии под действием любой входной буквы переходить в одно из нескольких возможных состояний. Разумеется, эта способность *недетерминированного перехода* имеет чисто воображаемый характер и в реальных автоматических устройствах не встречается. Однако при ее наличии автомат приобретает новые возможности, позволяющие в ряде случаев проще решать поставленные задачи. Вместе с тем класс множеств, допускаемых недетерминированными автоматами, не пополняется новыми (не конечно-автоматными) множествами.

Недетерминированный автомат $\mathcal{A}$, как и обычный конечный автомат, задается набором (A, Q, f, q_1, F), где A, Q, q_1, F имеют тот же смысл, что и ранее. Отличие состоит в функции переходов f. Для недетерминированного автомата функция f, вообще говоря, не является (однозначным) отображением множества $A \times Q$ в множество Q. Функцию f следует рассматривать как функцию, отображающую множество $A \times Q$ в множество

$2^Q \setminus \{\varnothing\}$ всех непустых подмножеств множества Q, либо как специальный вид соответствия между множествами $A \times Q$ и Q. Итак, для любой пары (a_i, q_j) значением функции f будем считать некоторое непустое подмножество множества Q.

Пусть $\bar{a} = a_{i_1} a_{i_2} \dots a_{i_n}$ — слово в алфавите A. В недетерминированном автомате $\mathcal{A}$ этому слову соответствует, вообще говоря, несколько последовательностей состояний длины n (путей длины n в диаграмме Мура). Более точно, пусть, например, $f(a_{i_1}, q_1) = \{q_{j_1}, \dots, q_{j_s}\}$, где $s \geqslant 1$. Тогда под действием буквы a_{i_1} автомат $\mathcal{A}$ из состояния q_1 может перейти в любое из состояний $q_{j_1}, \dots, q_{j_s}$. Предположим далее, что

$$f(a_{i_2}, q_{j_1}) = \{q_{k_1}, \dots, q_{k_t}\}, \dots, f(a_{i_2}, q_{j_s}) = \{q_{l_1}, \dots, q_{l_u}\}.$$

Тогда под действием слова $a_{i_1} a_{i_2}$ автомат $\mathcal{A}$ может перейти в любое из состояний $q_{k_1}, \dots, q_{k_t}, \dots, q_{l_1}, \dots, q_{l_u}$ (отметим, что множества $\{q_{k_1}, \dots, q_{k_t}\}$, $\dots, \{q_{l_1}, \dots, q_{l_u}\}$ могут пересекаться). Вообще, если автомат $\mathcal{A}$ под действием слова $a_{i_1} \dots a_{i_d}$ (где $d < n$) может перейти в состояние q_j и

$$f(a_{i_{d+1}}, q_j) = \{q_{m_1}, \dots, q_{m_v}\},$$

то под действием слова $a_{i_1} \dots, a_{i_d} a_{i_{d+1}}$ автомат $\mathcal{A}$ может перейти в любое из состояний $q_{m_1}, \dots, q_{m_v}$.

Считаем, что автомат $\mathcal{A}$ *допускает* слово $\bar{a}$, если под действием слова $\bar{a}$ автомат $\mathcal{A}$ может попасть хотя бы в одно из состояний множества F. Так же, как и выше, через $D(\mathcal{A})$ обозначаем множество всех слов, допускаемых автоматом $\mathcal{A}$.

Теорема 2.4. *Класс множеств, допускаемых конечными недетерминированными автоматами, совпадает с классом конечно-автоматных множеств.*

Доказательство. В одну сторону утверждение теоремы очевидно: всякий конечный автомат можно считать недетерминированным автоматом и потому всякое конечно-автоматное множество допускается также и недетерминированным автоматом. Докажем утверждение теоремы в другую сторону.

Пусть $\mathcal{A} = (A, Q, f, q_1, F)$ — недетерминированный автомат и множество Q состоит из r состояний. Построим по нему обычный (детерминированный) автомат $\mathcal{A}'$, который допускает то же самое множество $D(\mathcal{A})$. Из анализа действия слова (из A^*) на недетерминированный автомат $\mathcal{A}$ видно, что в принципе любое непустое подмножество множества Q может являться множеством тех состояний, в каждое из которых автомат

$\mathcal{A}$ попадает под действием одного и того же входного слова. Поэтому в качестве состояний автомата $\mathcal{A}'$ возьмем все непустые подножества множества Q. Число таких подмножеств есть $r' = 2^r - 1$. Обозначим эти множества-состояния автомата $\mathcal{A}'$ через $q'_1, \ldots, q'_{r'}$.

Довольно понятно, как определить функцию переходов f' автомата $\mathcal{A}'$: если $q'_j = \{q_{j_1}, \ldots, q_{j_s}\}$, то, считая значения $f(a_i, q_{j_1}), \ldots, f(a_i, q_{j_s})$ подмножествами множества Q, положим

$$f'(a_i, q'_j) = q'_l, \quad \text{где} \quad q'_l = f(a_i, q_{j_1}) \cup \ldots \cup f(a_i, q_{j_s}).$$

Соответственно, множество F' автомата $\mathcal{A}'$ будет состоять из всех подмножеств множества Q, которые имеют хотя бы один общий элемент с множеством F. Начальным состоянием автомата $\mathcal{A}'$ является одноэлементное множество $\{q_1\}$.

Таким образом, для произвольного слова $\bar{a}$ в алфавите A автомат $\mathcal{A}'$, начиная с одноэлементного множества $\{q_1\}$, под действием слова $\bar{a}$ достигает состояния $\{q_{j_1}, \ldots, q_{j_s}\}$ тогда и только тогда, когда автомат $\mathcal{A}$ под действием слова $\bar{a}$ может попасть в любое из состояний $q_{j_1}, \ldots, q_{j_s}$. Отсюда сразу следует, что $D(\mathcal{A}') = D(\mathcal{A})$. Теорема доказана.

УПРАЖНЕНИЯ

7. Обобщим понятие недетерминированного автомата в двух направлениях. Во-первых, разрешим автомату вместо одного начального состояния иметь несколько начальных состояний. При этом множество, допустимое автоматом, будет состоять из всех слов, под действием которых автомат из какого-либо начального состояния переходит в какое-либо заключительное состояние. Во-вторых, не будем требовать, чтобы для любой буквы a_i входного алфавита и любого состояния q_j значение $f(a_i, q_j)$ было определено (в диаграмме Мура такого автомата не из всякого состояния q_j выходит дуга, помеченная буквой a_i). Назовем такие обобщенные недетерминированные автоматы *источниками*.

Доказать, что всякое множество, допустимое источником, является конечно-автоматным.

§ 4. Операции произведения и итерации

В § 1 мы определили операцию конкатенации слов. Нетрудно видеть, что операция конкатенации ассоциативна, но не коммутативна. По су-

ществу конкатенация является (некоммутативной) операцией умножения слов. На основе операции конкатенации дадим определение операции произведения множеств, состоящих из слов.

Пусть X, Y — произвольные непустые множества слов. *Произведением* множеств X, Y называется множество $X \cdot Y$ всех слов вида $\bar{a}\bar{b}$, где $\bar{a} \in X$ и $\bar{b} \in Y$. Иначе говоря,

$$X \cdot Y = \bigcup_{\bar{a} \in X,\ \bar{b} \in Y} \bar{a}\bar{b}.$$

Поскольку в основе операции произведения лежит операция конкатенации, операция произведения также будет ассоциативной, но не коммутативной операцией. Если $n \geqslant 1$, то вместо $X \cdot X \cdot \ldots \cdot X$ (n раз), как обычно, будем писать X^n. Удобно считать, что для любого непустого множества X множество X^0 определено и состоит из одного пустого слова. Отметим также, что для любого множества X справедливы равенства $\varnothing \cdot X = X \cdot \varnothing = \varnothing$.

Операция *итерации* — более сложная операция, которая основана на операциях произведения и объединения. Именно, если X — непустое множество слов, то *итерацией* X^* множества X называется бесконечное объединение

$$X^0 \cup X^1 \cup X^2 \cup \ldots \cup X^n \cup \ldots.$$

Иначе говоря, слово $\bar{a}$ принадлежит множеству X^* в том и только том случае, когда либо $\bar{a} = \Lambda$, либо существуют такие слова $\bar{a}_1, \ldots, \bar{a}_n$ из X (здесь n также является параметром), что $\bar{a} = \bar{a}_1 \ldots \bar{a}_n$. Заметим, что если в множество X входит непустое слово $\bar{a}$, то множеству X^* принадлежат все слова $\bar{a}, \bar{a}^2, \bar{a}^3, \ldots$, т.е. множество X^* является бесконечным. Напротив, если $X = \{\Lambda\}$, то $X^* = \{\Lambda\}$. Итерация пустого множества по определению есть пустое множество.

Обратим внимание на то, что множество A^* всех слов в алфавите A (включая пустое слово) действительно есть итерация множества A.

Цель этого параграфа — показать, что операции произведения и итерации не выводят за пределы класса конечно-автоматных множеств.

Теорема 2.5. *Операция произведения сохраняет конечную автоматность множеств.*

Доказательство. Пусть $\mathcal{A} = (A, Q, f, q_1, F)$, $\mathcal{B} = (A, Q', f', q'_1, F')$ — конечные автоматы и $X = D(\mathcal{A})$, $Y = D(\mathcal{B})$. Построим автомат $\mathcal{C}$, который допускает множество $X \cdot Y$.

Исходя из определения множества $X \cdot Y$, попытаемся в построении автомата $\mathcal{C}$ реализовать следующую идею: к каждому заключительному состоянию автомата $\mathcal{A}$ «присоединим» автомат $\mathcal{B}$ так, чтобы, попав в это заключительное состояние автомата $\mathcal{A}$, автомат $\mathcal{C}$ мог бы продолжать работать далее как автомат $\mathcal{B}$. Сразу ясно, что искомый автомат $\mathcal{C}$, реализующий эту идею, должен быть недетерминированным автоматом. Однако это не может служить препятствием на пути выполнения задуманного плана, поскольку в предыдущем параграфе указан способ «детерминизации» недетерминированного автомата.

Имеется еще одна техническая трудность, которую сразу можно и не заметить. Дело в том, что соединение автоматов $\mathcal{A}$ и $\mathcal{B}$ в заключительных состояниях автомата $\mathcal{A}$ в принципе «проходимо» в обоих направлениях: как от автомата $\mathcal{A}$ к автомату $\mathcal{B}$ (к чему мы стремимся), так и через начальное состояние автомата $\mathcal{B}$ — от автоматов $\mathcal{B}$ к автомату $\mathcal{A}$ (что может вызвать нежелательный эффект, несовместимый с определением множества $X \cdot Y$). Поэтому сначала мы должны несколько «модифицировать» автомат $\mathcal{B}$, чтобы исключить упомянутые переходы от автомата $\mathcal{B}$ к автомату $\mathcal{A}$.

По существу для этого автомат $\mathcal{B}$ должен удовлетворять только одному требованию: в нем не должно быть переходов в начальное состояние q'_1 (т.е. функция f' не должна принимать значение q'_1 ни для каких значений аргументов). Если автомат $\mathcal{B}$ изначально удовлетворяет этому требованию, то переходим к следующему этапу построения автомата $\mathcal{C}$. В противном случае вводим новое начальное состояние q'_{11}, не принадлежащее множеству Q'. Функцию f' доопределяем для состояния q'_{11} следующим образом:

$$f'(a_i, q'_{11}) = f'(a_i, q'_1) \quad (1 \leqslant i \leqslant k).$$

Если в автомате $\mathcal{B}$ состояние q'_1 входит в множество F' (и только в этом случае), то присоединим состояние q'_{11} к множеству F'. Это присоединение позволяет сохранить в качестве (возможного) допустимого слова пустое слово Λ.

Нетрудно убедиться в том, что так «модифицированный» автомат $\mathcal{B}$ будет допускать то же самое множество Y. Кроме того, — и это есть главная причина «модификации» автомата $\mathcal{B}$ — в новом автомате отсутствуют переходы в начальное состояние q'_{11}. Чтобы далее не вводить дополнительных обозначений, будем предполагать, что с самого начала автомат $\mathcal{B}$ удовлетворяет сформулированному выше требованию.

Теперь мы готовы к тому, чтобы в деталях определить автомат $\mathcal{C}$. Пусть

$$F = \{q_{j_1}, \ldots, q_{j_s}\}, \quad Q' = \{q'_1, \ldots, q'_t\}.$$

Множество состояний автомата $\mathcal{C}$ будет представлять собой объединение $Q \cup Q'$. Поскольку мы хотим «присоединить» автомат $\mathcal{B}$ своим начальным состоянием q'_1 к состояниям $q_{j_1}, \ldots, q_{j_s}$ автомата $\mathcal{A}$, можно просто считать, что состояния $q_{j_1}, \ldots, q_{j_s}$ совпадают с состоянием q'_1.

Далее нам необходимо определить функцию переходов h автомата $\mathcal{C}$. Пусть q — произвольное состояние автомата $\mathcal{C}$. Если q не входит в множество F (т.е. отлично от состояний $q_{j_1}, \ldots, q_{j_s}$), то в соответствии с планом построения автомата $\mathcal{C}$ значение $h(a_i, q)$ должно совпадать с соответствующим значением $f(a_i, q)$ или $f'(a_i, q)$ (в зависимости от того, в какое из множеств Q или Q' входит состояние q). Пусть теперь $q \in F$ и, например, $q = q_{j_v}$. Тогда функция h на наборе (a_i, q) принимает два значения $f(a_i, q_{j_v})$ и $f'(a_i, q'_1)$.

Таким образом, недетерминированность автомата $\mathcal{C}$ проявляется только в состояниях $q_{j_1}, \ldots, q_{j_s}$. В них автомат $\mathcal{C}$ может продолжать работать как автомат $\mathcal{A}$ (переходы вида $f(a_i, q_{j_v})$), либо начать работать как автомат $\mathcal{B}$ (переходы вида $f'(a_i, q'_1)$. Заключительными состояниями автомата $\mathcal{C}$ объявим все состояния из множества F'. Просматривая теперь все этапы построения автомата $\mathcal{C}$, убеждаемся в том, что автомат $\mathcal{C}$ действительно допускает множество $X \cdot Y$. Теорема доказана.

Теорема 2.6. *Операция итерации сохраняет конечную автоматность множеств.*

Доказательство. Пусть $\mathcal{A} = (A, Q, f, q_1, F)$ — конечный автомат и $X = D(\mathcal{A})$. Если $X = \{\Lambda\}$, то $X^* = \{\Lambda\}$ и автомат $\mathcal{A}$ допускает множество X^*. Предположим далее, что множество X содержит хотя бы одно непустое слово. По автомату $\mathcal{A}$ будем строить недетерминированный автомат $\mathcal{C}$, который допускает множество X^*. Обращаясь к определению множества X^*, видим, что автомат $\mathcal{C}$ мог бы получиться из автомата $\mathcal{A}$, если в автомате $\mathcal{A}$ «отождествить» все заключительные состояния с начальным состоянием q_1.

Более формально: следующим образом расширим функцию переходов f автомата $\mathcal{A}$ до функции f' автомата $\mathcal{C}$: если состояние q_j входит в множество F, то пусть для любой буквы a_i из A значением $f'(a_i, q_j)$ будут состояния $f(a_i, q_j)$ и $f(a_i, q_1)$ (может статься, что $f(a_i, q_j) = f(a_i, q_1)$, тогда функция f' принимает на наборе (a_i, q_j) одно значение $f(a_i, q_j)$).

Начальное состояние q_1 и множество заключительных состояний F мы оставляем для автомата $\mathcal{C}$ без изменений.

Легко понять, как будет работать автомат $\mathcal{C}$ при подаче на его вход произвольного слова. До того момента, пока автомат $\mathcal{C}$ не попадет в одно из состояний множества F, он будет действовать точно так же, как автомат $\mathcal{A}$. При попадании в состояние множества F перед автоматом $\mathcal{C}$ возникает альтернатива: либо продолжать действовать как автомат $\mathcal{A}$ (т.е. воспользоваться переходом $f(a_i, q_j)$), либо принять данное заключительное состояние за начальное состояние автомата $\mathcal{A}$ и в соответствии с этим действовать как автомат $\mathcal{A}$, который находится в состоянии q_1 (т.е. использовать переход $f(a_i, q_1)$). Разумеется, подобные альтернативы для рассматриваемого входного слова могут возникать не один раз.

Таким образом, всякое слово, допускаемое автоматом $\mathcal{C}$, представляет собой произведение нескольких слов, допускаемых автоматом $\mathcal{A}$. Верно и обратное: всякое слово, представимое в виде произведения нескольких слов, допускаемых автоматом $\mathcal{A}$, допускается автоматом $\mathcal{C}$. Значит, $D(\mathcal{C})$ совпадает с множеством

$$X^1 \cup X^2 \cup \ldots \cup X^n \cup \ldots \tag{2.3}$$

Если $\Lambda \in X$, то множество (2.3) равно множеству X^*. В противном случае, чтобы получить множество X^*, необходимо к множеству (2.3) добавить пустое слово. Однако нам известно, что одноэлементное множество $\{\Lambda\}$ является конечно-автоматным. Применяя теперь последовательно теорему 2.4 к множеству $D(\mathcal{C})$ и теорему 2.3 к множествам $D(\mathcal{C})$ и $\{\Lambda\}$, получаем конечную автоматность множества X^*. Теорема доказана.

УПРАЖНЕНИЯ

8. Отправляясь от множеств $\{a_1\}$ и $\{a_2\}$, построить с помощью операций объединения, произведения и итерации конечно-автоматное множество из задачи 2е.

9. Пусть $\bar{a}$ — произвольное слово в алфавите A. Сколько раз нужно применить операцию итерации, чтобы получить множество $A^* \setminus \{\bar{a}\}$ из множеств $\{a_1\}, \ldots, \{a_k\}$ с помощью операций объединения, произведения и итерации?

§ 5. Регулярные множества. Теорема Клини

Как мы видели в § 2, конечно-автоматные множества можно определять без использования конечных автоматов. Еще одно индуктивное определение конечно-автоматных множеств предложено С. Клини.

Введем понятия *регулярного выражения* над алфавитом A и параллельно — определяемого им *регулярного множества* в алфавите A.

1. $\varnothing$ является регулярным выражением над алфавитом A и задает пустое регулярное множество.

2. Λ является регулярным выражением над алфавитом A и задает регулярное множество $\{\Lambda\}$, состоящее из одного пустого слова.

3. Если a_i — буква алфавита A, то символ a_i является регулярным выражением над алфавитом A и задает регулярное множество $\{a_i\}$, состоящее из однобуквенного слова a_i.

4. Пусть α, β — регулярные выражения над алфавитом A, которые задают регулярные множества X, Y слов в алфавите A. Тогда

$$(\alpha \cup \beta), \quad (\alpha\beta), \quad (\alpha)^*$$

суть регулярные выражения над алфавитом A, которые задают соответственно регулярные множества $X \cup Y$, $X \cdot Y$ и X^*.

При записи регулярных выражений мы будем иногда опускать скобки, считая, что операция $*$ имеет более высокий приоритет, чем операции $\cup$ и $\cdot$, а операция $\cdot$ — более высокий приоритет, чем операция $\cup$.

Из § 1 нам известно, что множества $\varnothing$, $\{\Lambda\}$ и $\{a_i\}$ являются конечно-автоматными, а из §§ 2,4 — что операции объединения, произведения и итерации сохраняют конечную автоматность множеств. Тем самым мы доказали в одну сторону следующую теорему.

Теорема 2.7 (С. Клини). *Класс конечно-автоматных множеств совпадает с классом регулярных множеств.*

Докажем теперь, что всякое конечно-автоматное множество является регулярным. Пусть $\mathcal{A} = (A, Q, f, q_1, F)$ — конечный автомат, который допускает множество X, $Q = \{q_1, \ldots, q_r\}$ и $F = \{q_{j_1}, \ldots, q_{j_s}\}$. Покажем, что множество X регулярно.

Заметим сначала, что множество F заключительных состояний можно считать одноэлементным. В самом деле, как вытекает из определения допустимого множества, множество X есть объединение s попарно

не пересекающихся множеств $X_1, \ldots, X_s$, которые допускаются соответственно автоматами

$$\mathcal{A}_1 = (A, Q, f, q_1, \{q_{j_1}\}), \ldots, \mathcal{A}_s = (A, Q, f, q_1, \{q_{j_s}\})$$

(считаем, что все состояния $q_{j_1}, \ldots, q_{j_s}$ достижимы из начального состояния q_1).

Итак, пусть далее $F = \{q_t\}$, где $1 \leqslant t \leqslant r$. Для любого k $(0 \leqslant k \leqslant r)$ и любых i, j $(1 \leqslant i, j \leqslant r)$ обозначим через Z_{ij}^k множество всех слов в алфавите A, которые переводят автомат $\mathcal{A}$ из состояния q_i в состояние q_j, используя при этом в качестве «промежуточных» только состояния $q_1, \ldots, q_k$. Иными словами, множество Z_{ij}^k состоит из всех слов $a_{i_1} \ldots a_{i_n}$, для которых найдутся такие состояния $q_{l_2}, \ldots, q_{l_n}$ из множества $\{q_1, \ldots, q_k\}$, что

$$f(a_{i_1}, q_i) = q_{l_2},\ f(a_{i_2}, q_{l_2}) = q_{l_3}, \ldots, f(a_{i_{n-1}}, q_{l_{n-1}}) = q_{l_n},\ f(a_{i_n}, q_{l_n}) = q_j.$$

Далее мы покажем индукцией по k, что все множества Z_{ij}^k регулярны. Поскольку, очевидно, $Z_{1t}^r = X$, отсюда будет следовать регулярность множества X.

Начнем с множеств Z_{ij}^0. По определению множество Z_{ij}^0 состоит из всех слов в алфавите A, которые переводят автомат $\mathcal{A}$ из состояния q_i в состояние q_j и при этом не используются никакие «промежуточные» состояния.

Пусть $i = j$. Тогда множество Z_{ii}^0 заведомо включает в себя пустое слово. Кроме того, если есть буквы $a_{m_1}, \ldots, a_{m_p}$ алфавита A, переводящие состояние q_i в себя, то, очевидно, множество Z_{ii}^0 состоит из пустого слова и этих букв, т.е. $Z_{ii}^0 = \{\Lambda, a_{m_1}, \ldots, a_{m_p}\}$. Итак, множество Z_{ii}^0 либо состоит из одного пустого слова, либо из слов $\Lambda, a_{m_1}, \ldots, a_{m_p}$. В обоих случаях множество Z_{ii}^0 конечно и потому регулярно.

Пусть $i \neq j$. Тогда множество Z_{ij}^0 либо пусто (если отсутствуют буквы, переводящие состояние q_i непосредственно в состояние q_j), либо состоит из некоторых букв алфавита A. Понятно, что в обоих случаях множество Z_{ij}^0 также регулярно.

Предположим, что регулярность множеств Z_{ij}^{k-1} установлена для всех i, j $(1 \leqslant i, j \leqslant r)$ и некоторого k, где $1 \leqslant k \leqslant r$. Докажем регулярность всех множеств Z_{ij}^k.

Пусть $\bar{a}$ — произвольное слово, принадлежащее множеству Z_{ij}^k. Тогда под действием слова $\bar{a}$ автомат $\mathcal{A}$ переходит из состояния q_i в состояние q_j. Если при этом не используется промежуточное состояние q_k, то

$\bar{a} \in Z_{ij}^{k-1}$. В противном случае слово $\bar{a}$ можно представить в виде конкатенации трех слов $\bar{a}_1, \bar{a}_2, \bar{a}_3$, где:

1) под действием слова $\bar{a}_1$ автомат $\mathcal{A}$ переходит из состояния q_i в состояние q_k, используя в качестве промежуточных только состояния $q_1, \ldots, q_{k-1}$ (т.е. $\bar{a}_1 \in Z_{ik}^{k-1}$);

2) под действием слова $\bar{a}_2$ автомат $\mathcal{A}$ совершает несколько циклических переходов из состояния q_k в себя, используя в каждом цикле в качестве промежуточных только состояния $q_1, \ldots, q_{k-1}$ (т.е. $\bar{a}_2 \in (Z_{kk}^{k-1})^*$);

3) под действием слова $\bar{a}_3$ автомат $\mathcal{A}$ переходит из состояния q_k в состояние q_j, используя в качестве промежуточных только состояния $q_1, \ldots, q_{k-1}$ (т.е. $\bar{a}_3 \in Z_{kj}^{k-1}$).

В итоге мы приходим к формуле

$$Z_{ij}^k = Z_{ij}^{k-1} \cup Z_{ik}^{k-1} \cdot (Z_{kk}^{k-1})^* \cdot Z_{kj}^{k-1},$$

которая показывает, что в случае регулярности множеств $Z_{ij}^{k-1}, Z_{ik}^{k-1}, Z_{kk}^{k-1}, Z_{kj}^{k-1}$ множество Z_{ij}^k также будет регулярным. Это завершает доказательство теоремы Клини.

УПРАЖНЕНИЯ

10. Используя определение регулярного множества, доказать регулярность следующих множеств в алфавите $\{a_1, a_2\}$:
a) любое конечное множество слов; b) дополнение (до множества $\{a_1, a_2\}^*$) к любому конечному множеству слов; c) множество всех слов, составленных из слов $\bar{a}_1 \ldots \bar{a}_n$; d) множество всех слов, содержащих в качестве подслова заданное слово $\bar{a}$; e) множество всех слов, которые не содержат в качестве подслова заданное слово $\bar{a}$.

11. Рассмотрим следующее обобщение конечного автомата. Разрешим переход из одного состояния в следующее состояние обобщенного автомата не только под действием букв входного алфавита (как это предусматривается в обычном автомате), но также и под действием произвольных слов во входном алфавите. Кроме того, разрешим автомату под действием одного и того же слова переходить в несколько различных состояний (недетерминированность автомата). Таким образом, функция переходов f обобщенного автомата $\mathcal{A} = (A, Q, f, q_1, F)$ является *частичной* функцией, заданной на некотором конечном подмножестве множества $A^* \times Q$, которая некоторым парам $(\bar{a}, q_j)$ сопоставляет непустые подмножества множества Q.

Доказать, что всякое множество, допустимое таким обобщенным автоматом, является конечно-автоматным.

12. Определим операцию подстановки нескольких множеств слов в множество слов. Пусть имеются алфавиты $A = \{a_1, \ldots, a_k\}$ и B, произвольное множество X слов в алфавите A и произвольные непустые множества $Y_1, \ldots, Y_k$ слов в алфавите B. Подстановкой множеств $Y_1, \ldots, Y_k$ в множество X будем называть множество $X[Y_1, \ldots, Y_k]$ всех слов в алфавите B, которые получаются по следующему правилу: берется произвольное слово $a_{i_1} \ldots a_{i_n}$ из X и в множество $X[Y_1, \ldots, Y_k]$ заносятся все слова из множества $Y_{i_1} \cdot \ldots \cdot Y_{i_n}$. Если множеству X принадлежит пустое слово, то «подстановка» множеств $Y_1, \ldots, Y_k$ в пустое слово дает пустое слово. Если $X = \varnothing$, то по определению считаем, что $X[Y_1, \ldots, Y_k] = \varnothing$.

Доказать, что подстановка конечно-автоматных множеств в конечно-автоматное множество дает конечно-автоматное множество.

13. Обращением слова $a_{i_1} \ldots a_{i_n}$ называется слово $a_{i_n} \ldots a_{i_1}$. Обращение множества слов X — это множество, состоящее из обращений всех слов из X.

Доказать, что обращение любого конечно-автоматного множества является конечно-автоматным множеством.

Глава 3

КОНЕЧНЫЕ АВТОМАТЫ-ПРЕОБРАЗОВАТЕЛИ

§ 1. Конечный автомат с выходом. Теорема Мура

Расширим понятие конечного автомата. Пусть $A = \{a_1, \ldots, a_k\}, B = \{b_1, \ldots, b_m\}$ — конечные алфавиты. Мы хотим определить конечный автомат с входным алфавитом A и выходным алфавитом B так, чтобы он мог вычислять словарную функцию вида $A^* \to B^*$, т.е. отображение множества A^* всех слов в алфавите A в множество B^* всех слов в алфавите B. Для реализации этого намерения проще всего снабдить автомат без выхода так называемой *функцией выходов*, которая на каждом шаге работы автомата выдает некоторую букву выходного алфавита B. Отметим, что в концепции автомата с выходом заключительные состояния не предусмотрены.

Итак, конечный автомат $\mathcal{A}$ с выходом задается набором (A, B, Q, f, g, q_1), где A — входной, B — выходной алфавиты автомата, $Q = \{q_1, \ldots, q_r\}$ — множество состояний, f — функция переходов, отображающая множество $A \times Q$ в множество Q, g — функция выходов, отображающая множество $A \times Q$ в множество B, q_1 — начальное состояние автомата $\mathcal{A}$.

Считаем, что автомат $\mathcal{A}$ преобразует (переводит) слово $a_{i_1} \ldots a_{i_n}$ в алфавите A в соответствующее слово $b_{j_1} \ldots b_{j_n}$ в алфавите B согласно следующему правилу. Полагаем $b_{j_1} = g(a_{i_1}, q_1)$. Находим значение $q_{s_1} = f(a_{i_1}, q_1)$. Тогда $b_{j_2} = g(a_{i_2}, q_{s_2})$. Далее вычисляем значение $f(a_{i_2}, q_{s_1})$. Вообще, если уже найдено, что $q_{s_l} = f(a_{i_l}, q_{s_{l-1}})$, то полагаем $b_{j_{l+1}} = g(a_{i_{l+1}}, q_{s_l})$ и находим значение $q_{s_{l+1}} = f(a_{i_{l+1}}, q_{s_l})$.

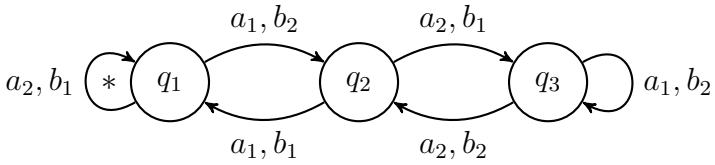
По определению считаем, что пустое слово автомат $\mathcal{A}$ переводит также в пустое слово. Тем самым автомат $\mathcal{A}$ задает отображение множества A^* в множество B^*, которое мы будем называть *конечно-автоматной функцией*, реализуемой (вычислимой) автоматом $\mathcal{A}$. Для упрощения записи эту функцию будем обозначать тем же символом $\mathcal{A}$.

Из описания вычисления функции $\mathcal{A}$ легко усмотреть, что с введением параметра t («время») эту функцию можно представить с помощью следующих *канонических уравнений*:

$$\begin{cases} y(t) = f(x(t), q(t-1)), \\ q(t) = g(x(t), q(t-1)), \\ q(0) = q_1, \end{cases}$$

где через $y(t)$ обозначен выходной символ автомата $\mathcal{A}$ в момент времени t.

Так же, как для автомата без выхода, автомат $\mathcal{A}$ и реализуемую им конечно-автоматную функцию можно задать с помощью диаграммы Мура. Единственное отличие для автоматов с выходом состоит в том, что на дуге диаграммы Мура, ведущей из состояния q_j в состояние q_l и помеченной буквой a_i (т.е. при выполнении условия $f(a_i, q_j) = q_l$), вслед за буквой a_i через запятую ставится буква $b_p = g(a_i, q_j)$. Ниже на рисунке изображен пример диаграммы Мура для автомата с входным алфавитом $A = \{a_1, a_2\}$ и выходным алфавитом $B = \{b_1, b_2\}$.



Для автоматов с выходом одной из центральных задач является проблема *эквивалентности* автоматов. Назовем два автомата с выходом *эквивалентными*, если совпадают функции, реализуемые этими автоматами. Пусть $\mathcal{A} = (A, B, Q, f, g, q_1)$ — автомат с выходом, q_j, q_l — состояния автомата $\mathcal{A}$. Будем говорить, что состояния q_j, q_l *эквивалентны* (в автомате $\mathcal{A}$), если эквивалентны автоматы

$$\mathcal{A}_j = (A, B, Q, f, g, q_j), \quad \mathcal{A}_l = (A, B, Q, f, g, q_l).$$

Неэквивалентные состояния называем также *отличимыми* состояниями. Если состояния q_j, q_l отличимы в автомате $\mathcal{A}$, то существует такое слово $\bar{a}$ в алфавите A, что $\mathcal{A}_j(\bar{a}) \neq \mathcal{A}_l(\bar{a})$. В этом случае говорят, что слово $\bar{a}$ *отличает* состояния q_j, q_l в автомате $\mathcal{A}$.

В нижеследующей теореме дается неулучшаемая верхняя оценка для длины слова, отличающего в автомате с r состояниями два отличимых состояния.

Теорема 3.1 (Э. Мур). *Если в автомате с $r \geqslant 2$ состояниями два состояния отличимы, то они отличимы словом длины, не превосходящей $r - 1$.*

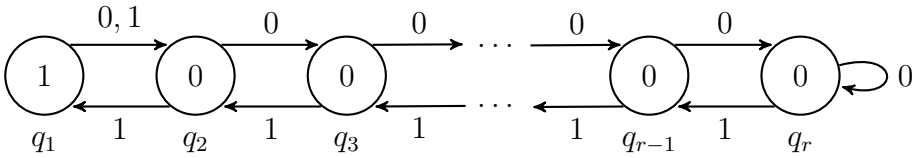
Доказательство. Пусть $\mathcal{A} = (A, B, Q, f, g, q_1)$ — автомат с выходом, имеющий r состояний. Для любого $n \geqslant 0$ определим на множестве Q отношение эквивалентности ε_n: состояния $q_j, q_l \in Q$ считаем эквивалентными в смысле отношения ε_n в том и только том случае, когда

q_j, q_l неотличимы в автомате $\mathcal{A}$ словами длины n (очевидно, что также и словами меньшей длины). Словом длины 0 (т.е. пустым словом) будут неотличимы любые состояния из Q. Иначе гоаоря, ε_0 — это *полное* отношение эквивалентности на Q (все состояния из Q эквивалентны в смысле отношения ε_0).

При переходе от отношения ε_n к отношению ε_{n+1} состояния, неэквивалентные в смысле ε_n, будут, конечно, неэквивалентны и в смысле отношения ε_{n+1}. Пусть q_j, q_l — состояния, эквивалентные в смысле отношения ε_n, но отличимые в автомате $\mathcal{A}$. Предположим, что слово $a_{i_1} \dots a_{i_p}$ отличает состояния q_j, q_l в автомате $\mathcal{A}$. Будем также считать, что длина p выбрана наименьшей возможной (т.е. словами длины $p-1$ состояния q_j, q_l отличить невозможно). Понятно, что $p \geqslant n+1$. Обозначим через q_u, q_v состояния, в которые переходят автоматы $\mathcal{A}_j, \mathcal{A}_l$ под действием слова $a_{i_1} \dots a_{i_{p-n-1}}$. Тогда состояния q_u, q_v отличимы словом $a_{i_{p-n}} \dots a_{i_p}$ длины $n+1$, но неотличимы словами длины n (иначе состояния q_j, q_l были бы отличимы словами длины $p-1$, что невозможно по выбору числа p). Иначе гоаоря, состояния q_u, q_v эквивалентны в смысле отношения ε_n, но не эквивалентны в смысле отношения ε_{n+1}.

Итак, если какое-либо множество состояний, эквивалентных в смысле отношения ε_n, содержит отличимые состояния, то при переходе к отношению ε_{n+1} хотя бы одно из таких множеств разбивается на два или более подмножеств, эквивалентных в смысле отношения ε_{n+1}. Поскольку для отношения ε_0 это множество одно, а всего число таких множеств не превосходит r, получаем, что после отношения ε_{r-1} дальнейшего «дробления» отношений эквивалентности заведомо не происходит. Другими словами, любые отличимые в автомате $\mathcal{A}$ состояния отличимы словом длины, не превосходящей $r-1$. Теорема доказана.

Как видно из приводимого ниже примера, верхняя оценка, указанная в теореме 3.1, в общем случае не может быть понижена.



Состояния q_{r-1}, q_r в этом примере отличимы словом 1^{r-1}, но не отличимы никаким словом меньшей длины (выходной символ автомата не зависит от входа и указан внутри соответствующего кружка).

Итак, если автомат $\mathcal{A}$ имеет r состояний, то, как вытекает из доказанной теоремы, для проверки отличимости произвольных двух состояний q_j, q_l автомата $\mathcal{A}$ достаточно подать на вход автоматов $\mathcal{A}_j, \mathcal{A}_l$ все слова $\bar{a}$ длины, не превосходящей $r-1$, и сравнить получающиеся при этом выходы $\mathcal{A}_j(\bar{a})$ и $\mathcal{A}_l(\bar{a})$. Разумеется, использование здесь автоматов $\mathcal{A}_j, \mathcal{A}_l$ вместо автомата $\mathcal{A}$ — не более чем переформулирование задачи в удобных терминах. На самом деле автомат $\mathcal{A}$ «приводится» в состояние q_j (или состояние q_l), а затем на вход автомата подаются все слова $\bar{a}$ указанной длины. Стоит еще отметить, что, как показывает приведенный выше пример автомата с выходом, без дополнительной информации об автомате (помимо информации о числе состояний автомата) уменьшить число слов, подаваемых на вход автомата, нельзя.

На основе теоремы 3.1 можно решить вопрос об эквивалентности двух автоматов. Именно, пусть известно, что автоматы

$$\mathcal{A} = (A, B, Q, f, g, q_1), \quad \mathcal{A}' = (A, B, Q', f', g', q_1')$$

имеют соответственно r и r' состояний. Считаем, что множества Q и Q' не имеют общих элементов. Для проверки эквивалентности автоматов $\mathcal{A}, \mathcal{A}'$ образуем формальное объединение $\mathcal{A} \cup \mathcal{A}'$ автоматов $\mathcal{A}$ и $\mathcal{A}'$. Именно, множеством состояний этого объединения объявим множество $Q \cup Q'$. Функции переходов и выходов будут «составлены» из функций f, g и f', g', действующие каждая на своем множестве $A \times Q$ или $A' \times Q'$. Выбор начального состояния в автомате $\mathcal{A} \cup \mathcal{A}'$ роли не играет. Тогда отличимость состояний q_1, q_1' в автомате $\mathcal{A} \cup \mathcal{A}'$ равносильна неэквивалентности автоматов $\mathcal{A}, \mathcal{A}'$. Согласно теореме 3.1, для выяснения отличимости состояний q_1, q_1' достаточно рассмотреть все слова длины, не превосходящей $r + r' - 1$. Если теперь вернуться к исходным автоматам $\mathcal{A}$ и $\mathcal{A}'$, то получим, что для проверки эквивалентности автоматов $\mathcal{A}, \mathcal{A}'$ достаточно подать на их входы все слова $\bar{a}$ длины, не превосходящей $r + r' - 1$, и сравнить получающиеся при этом выходы $\mathcal{A}(\bar{a})$ и $\mathcal{A}'(\bar{a})$.

УПРАЖНЕНИЯ

1. Построить автомат с входным и выходным алфавитом $\{0, 1\}$, который переводит слова длины 1 и 2 соответственно в слова 0 и 00, а всякое слово $a_{i_1} \dots a_{i_n}$, где $n \geqslant 3$, — в слово $00a_{i_1} \dots a_{i_{n-2}}$.

§ 2. Остаточные функции. Вес функции

В главе 2 мы ввели понятия правоинвариантной эквивалентности и регулярного множества, что позволило дать описание конечно-автоматных множеств, не опирающееся на понятие автомата. Нечто подобное мы бы хотели получить для конечно-автоматных функций. Для этого нам потребуется ввести понятия детерминированной функции и остаточной функции.

Пусть A, B — конечные алфавиты, φ — функция, отображающая множество A^* в множество B^*. Назовем функцию φ *детерминированной*, если для любых слов $\bar{a}, \bar{b}$ в алфавите A слово $\varphi(\bar{a})$ является началом слова $\varphi(\bar{a}\bar{b})$ и, кроме того, $|\varphi(\bar{a})| = |\bar{a}|$, где $|\bar{a}|$ есть длина слова $\bar{a}$. Иными словами, функция φ детерминированна, если она сохраняет длину слова и для любого $\bar{a} \in A^*$ длины n i-я $(1 \leqslant i \leqslant n)$ буква слова $\varphi(\bar{a})$ зависит только от первых i букв слова $\bar{a}$.

Свойство детерминированности легко устанавливается для конечно-автоматных функций. В самом деле, пусть функция φ реализуется автоматом $\mathcal{A} = (A, B, Q, f, g, q_1)$. Тогда при подаче на вход автомата $\mathcal{A}$ слова $\bar{a}\bar{b}$ (буквы слова $\bar{a}\bar{b}$ подаются на вход автомата $\mathcal{A}$ слева направо, начиная с первой буквы слова $\bar{a}$) автомат $\mathcal{A}$ сначала перерабатывает слово $\bar{a}$ в слово $\varphi(\bar{a})$, а затем слово $\bar{b}$ — в оставшуюся часть слова $\varphi(\bar{a}\bar{b})$. Таким образом, $\varphi(\bar{a})$ есть начало слова $\varphi(\bar{a}\bar{b})$. Свойство сохранения длины слова очевидно.

Для детерминированной функции φ введем понятие *остаточной функции*. Пусть $\bar{a}$ — произвольное слово в алфавите A. Согласно определению детерминированности для всякого слова $\bar{b}$ в алфавите A слово $\varphi(\bar{a})$ есть начало слова $\varphi(\bar{a}\bar{b})$. Следовательно, можно следующим образом определить *остаточную функцию* $\varphi_{\bar{a}}$, отображающую A^* в B^*: значением $\varphi_{\bar{a}}(\bar{b})$ пусть будет то (единственное) слово, конкатенация которого со словом $\varphi(\bar{a})$ дает слово $\varphi(\bar{a}\bar{b})$. Иначе говоря, $\varphi_{\bar{a}}(\bar{b})$ есть окончание слова $\varphi(\bar{a}\bar{b})$, которое в слове $\varphi(\bar{a}\bar{b})$ следует за началом $\varphi(\bar{a})$ (если $\varphi(\bar{a}) = \varphi(\bar{a}\bar{b})$, то, очевидно, $\varphi_{\bar{a}}(\bar{b}) = \Lambda$). Таким образом,

$$\varphi(\bar{a}\bar{b}) = \varphi(\bar{a})\varphi_{\bar{a}}(\bar{b}).$$

Отметим, что остаточная функция детерминированной функции φ также будет детерминированной функцией. В самом деле, пусть $\varphi_{\bar{a}}$ — остаточная функция функции φ. Возьмем произвольные слова $\bar{b}, \bar{c}$ в алфавите A. Согласно условию детерминированности функции φ слово

$\varphi(\bar{a}\bar{b})$ есть начало слова $\varphi(\bar{a}\bar{b}\bar{c})$. Вместе с тем по определению остаточной функции $\varphi_{\bar{a}}$ имеем

$$\varphi(\bar{a}\bar{b}) = \varphi(\bar{a})\varphi_{\bar{a}}(\bar{b}), \quad \varphi(\bar{a}\bar{b}\bar{c}) = \varphi(\bar{a})\varphi_{\bar{a}}(\bar{b}\bar{c}).$$

Следовательно, слово $\varphi_{\bar{a}}(\bar{b})$ есть начало слова $\varphi_{\bar{a}}(\bar{b}\bar{c})$ и $\varphi_{\bar{a}}$ — детерминированная функция.

Доказанный факт позволяет представлять значения детерминированной функции φ на словах $\bar{a} = \bar{a}_1\bar{a}_2\ldots\bar{a}_n$ в виде конкатенации значений остаточных функций функции φ на словах $\bar{a}_1, \ldots, \bar{a}_n$. Именно, по определению остаточной функции $\varphi_{\bar{a}_1}$ справедливо равенство

$$\varphi(\bar{a}_1\bar{a}_2\ldots\bar{a}_n) = \varphi(\bar{a}_1)\varphi_{\bar{a}_1}(\bar{a}_2\ldots\bar{a}_n).$$

Функция $\varphi_{\bar{a}_1}$ также является детерминированной. Поэтому будет выполняться равенство

$$\varphi_{\bar{a}_1}(\bar{a}_2\ldots\bar{a}_n) = \varphi_{\bar{a}_1}(\bar{a}_2)\psi_{\bar{a}_2}(\bar{a}_3\ldots\bar{a}_n),$$

где $\psi_{\bar{a}_2}$ — остаточная функция функции $\varphi_{\bar{a}_1}$. Однако, как легко следует из определения остаточной функции, $\psi_{\bar{a}_2} = \varphi_{\bar{a}_1\bar{a}_2}$. Значит,

$$\varphi_{\bar{a}_1}(\bar{a}_2\ldots\bar{a}_n) = \varphi_{\bar{a}_1}(\bar{a}_2)\varphi_{\bar{a}_1\bar{a}_2}(\bar{a}_3\ldots\bar{a}_n).$$

Продолжая этот процесс по индукции, придем к соотношению

$$\varphi(\bar{a}) = \varphi(\bar{a}_1\bar{a}_2\ldots\bar{a}_n) = \varphi(\bar{a}_1)\varphi_{\bar{a}_1}(\bar{a}_2)\varphi_{\bar{a}_1\bar{a}_2}(\bar{a}_3)\ldots\varphi_{\bar{a}_1\ldots\bar{a}_{n-1}}(\bar{a}_n). \qquad (3.1)$$

Назовем *весом* детерминированной функции число ее различных остаточных функций. Отметим, что вес детерминированной функции может быть бесконечным.

Для конечно-автоматных функций смысл остаточных функций вполне очевиден. Если функция φ реализуется автоматом $\mathcal{A}$, то функция $\varphi_{\bar{a}}$, как легко видеть, реализуется автоматом (A, B, Q, f, g, q_j), где q_j — то состояние автомата $\mathcal{A}$, в которое он переходит из состояния q_1 под действием входного слова $\bar{a}$. Из этого разъяснения сразу следует, что для конечного автомата $\mathcal{A}$ с r состояниями число остаточных функций не превосходит r. Таким образом, вес конечно-автоматной функции конечен и не превосходит числа состояний реализующего ее конечного автомата.

Обнаруженную связь между остаточными функциями и состояниями конечного автомата можно положить в основу определения конечно-автоматной функции через ее остаточные функции. Итак, пусть детерминированная функция $\varphi: A^* \to B^*$ имеет вес r. Выше мы условились

считать, что конечно-автоматная функция переводит пустое слово в пустое слово. Поэтому будем предполагать, что $\varphi(\Lambda) = \Lambda$. В этих предположениях определим автомат $\mathcal{A} = (A, B, Q, f, g, q_1)$ с r состояниями, который реализует функцию φ.

Пусть для функции φ все остаточные функции суть $\varphi_{\bar{a}_1}, \dots \varphi_{\bar{a}_r}$, где слова $\bar{a}_1, \dots, \bar{a}_r$ выбраны, например, с наименьшими возможными длинами. Можно считать, что $\bar{a}_1 = \Lambda$, и, следовательно, $\varphi_{\bar{a}_1} = \varphi$. Будем предполагать также, что состояния $q_1, \dots, q_r$ автомата $\mathcal{A}$ соответствуют словам $\bar{a}_1, \dots, \bar{a}_r$ (и функциям $\varphi_{\bar{a}_1}, \dots, \varphi_{\bar{a}_r}$). Функцию переходов f автомата $\mathcal{A}$ определим следующим образом: если $a_i \in A$, то $f(a_i, q_j) = q_l$, где $\varphi_{\bar{a}_l} = \varphi_{\bar{a}_j a_i}$. Из аналогичных соображений определяем функцию g: $g(a_i, q_j) = \varphi_{\bar{a}_j}(a_i)$.

Проверим, что так определенный автомат $\mathcal{A}$ действительно реализует функцию φ. Возьмем произвольное слово $\bar{a} = a_{i_1} \dots a_{i_n}$ в алфавите A. Если в соотношении (3.1) положить $\bar{a}_1 = a_{i_1}, \dots, \bar{a}_n = a_{i_n}$, то будем иметь

$$\varphi(\bar{a}) = \varphi(a_{i_1})\varphi_{a_{i_1}}(a_{i_2})\varphi_{a_{i_1}a_{i_2}}(a_{i_3}) \dots \varphi_{a_{i_1}\dots a_{i_{n-1}}}(a_{i_n}). \tag{3.2}$$

Однако в силу определения автомата $\mathcal{A}$ имеем

$$\varphi(a_{i_1}) = \varphi_{\bar{a}_1}(a_{i_1}) = g(a_{i_1}, q_1).$$

Далее, $\bar{a}_1 = \Lambda$ и $\bar{a}_1 a_{i_1} = a_{i_1}$. Пусть функция $\varphi_{a_{i_1}}$ имеет обозначение $\varphi_{\bar{a}_{j_2}}$. Тогда

$$f(a_{i_1}, q_1) = q_{j_2}, \quad \varphi_{a_{i_1}}(a_{i_2}) = g(a_{i_2}, q_{j_2}).$$

Вообще, предположим, что слова $a_{i_1} \dots a_{i_s}$ и $a_{i_1} \dots a_{i_s} a_{i_{s+1}}$ задают те же остаточные функции, что и слова $\bar{a}_{j_{s+1}}$ и $\bar{a}_{j_{s+2}}$ (напомним, что слову a_{i_1} у нас соответствует слово $\bar{a}_{j_2}$). Тогда в силу определения функций f, g будем иметь

$$f(a_{i_{s+1}}, q_{j_{s+1}}) = q_{j_{s+2}}, \quad \varphi_{a_{i_1}\dots a_{i_s}}(a_{i_{s+1}}) = g(a_{i_{s+1}}, q_{j_{s+1}}).$$

Таким образом, правую часть равенства (3.2) можно переписать в виде

$$g(a_{i_1}, q_1)g(a_{i_2}, q_{j_2}) \dots g(a_{i_n}, q_{j_n}),$$

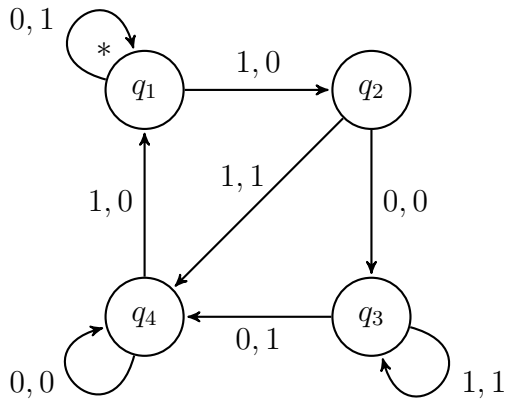
где состояния $q_{j_2}, \dots, q_{j_n}$ автомата $\mathcal{A}$ получаются из состояния q_1 последовательным применением функции переходов f. Иными словами, $\varphi(\bar{a})$ есть результат применения автомата $\mathcal{A}$ к слову $\bar{a}$.

Подведем итог наших исследований в виде теоремы.

Теорема 3.2. *Класс детерминированных функций конечного веса совпадает с классом конечно-автоматных функций.*

УПРАЖНЕНИЯ

2. Найти вес автоматной функции φ, заданной следующей диаграммой Мура:



§ 3. Конечные автоматы на сверхсловах

В этом и в следующих двух параграфах мы хотим рассмотреть конечные автоматы с выходом, которые оперируют с бесконечными последовательностями (сверхсловами). Переход к сверхсловам позволяет взглянуть на конечные автоматы-преобразователи с новой точки зрения, определить конечно-автоматные функции, близкие к непрерывным функциям действительных переменных, и установить конечную порождаемость класса конечно-автоматных функций.

Пусть A, B — конечные алфавиты. Обозначим через A^∞ множество всех счетно-бесконечных последовательностей (сверхслов), упорядоченных по типу натуральных чисел. Если алфавит A содержит не менее двух букв, то множество A^∞ состоит из континуального числа последовательностей (сверхслов). Сверхслово $a \in A^\infty$ будем записывать в виде $a = a(1)a(2)\ldots$, где $a(t) \in A$ при $t = 1, 2, \ldots$. Переменную x, принимающую значения в множестве A^∞, также записываем в виде $x = x(1)x(2)\ldots$, где переменная $x(t)$ принимают значения уже в множестве A.

Будем рассматривать функции $f(x_1, \ldots, x_n)$, отображающие множество $(A^\infty)^n$ в множество B^∞. Класс всех таких функций обозначим через $P^\infty_{A,B}$. Если алфавиты A и B совпадают, то вместо $P^\infty_{A,B}$ пишем P^∞_A. Для стандартного k-элементного алфавита $A = \{0, 1, \ldots, k-1\}$ вместо P^∞_A будем писать P^∞_k. По ряду причин функцию $f(x_1, \ldots, x_n)$ из класса $P^\infty_{A,B}$

удобно изображать с выходной переменной: $y = f(x_1, \ldots, x_n)$, где переменная y принимает значения в множестве B^∞.

Так же, как в § 2, первым приближением к конечно-автоматным функциям на сверхсловах будут детерминированные функции. Обозначим через X вектор переменных $x_1, \ldots, x_n$. Функцию $y = f(X)$ из класса $P^\infty_{A,B}$ назовем *детерминированной*, если для любых двух вектор-сверхслов a, b из $(A^\infty)^n$ и любого $t \geqslant 1$ из соотношений

$$c = f(a), \quad d = f(b), \quad a(1) = b(1), \ldots, a(t) = b(t)$$

следуют соотношения

$$c(1) = d(1), \ \ldots, \ c(t) = d(t).$$

Иными словами, при любом t значение $y(t)$ зависит только от значений $X(1), \ldots, X(t)$. Класс всех детерминированных функций из P^∞_k обозначим через $P_{\partial,k}$.

Нетрудно заметить, что введенное понятие детерминированной функции тесно связано с понятием детерминированной функции из § 2. Так, например, если $f(x)$ — детерминированная функция из § 2, то на ее основе легко создать детерминированную функцию типа $A^\infty \to B^\infty$, если для каждого сверхслова a из A^∞ последовательно вычислять значения функции f на началах $\bar{a}$ слова a. Обратно, если $f(x)$ — детерминированная функция типа $A^\infty \to B^\infty$, то значение функции f на сверхслове a можно естественным образом интерпретировать как значения детерминированной функции типа $A^* \to B^*$ на всех началах данного сверхслова a.

Из определения класса $P_{\partial,k}$ нетрудно вывести, что произвольная функция $f(x_1, \ldots, x_n)$ из $P_{\partial,k}$ полностью определяется последовательностью $\{F_1, F_2, \ldots\}$ функций k-значной логики, где

$$F_1 = F_1(x_1(1), \ldots, x_n(1)), \ F_2 = F_2(x_1(1), \ldots, x_n(1), x_1(2), \ldots, x_n(2)), \ldots$$

$$\ldots, F_t = F_t(x_1(1), \ldots, x_n(1), \ldots, x_1(t), \ldots, x_n(t)), \ldots.$$

Отсюда сразу следует, что при любом $k \geqslant 2$ и любом $n \geqslant 1$ число всех n-местных функций класса $P_{\partial,k}$ континуально.

Рассмотрим несколько примеров детерминированных функций из класса $P_{\partial,k}$.

а) Пусть $\varphi(x_1, \ldots, x_n)$ — произвольная функция k-значной логики. Определим детерминированную функцию $y = f_\varphi(x_1, \ldots, x)$ равенствами

$$y(t) = \varphi(x_1(t), \ldots, x_n(t)), \quad t = 1, 2, \ldots.$$

Функции вида f_φ носят название *истинностных* функций.

б) Зададим функцию $y = з(x)$ следующими равенствами:

$$y(1) = 0, \quad y(t) = x(t-1) \text{ при } t \geqslant 2.$$

Функция $з(x)$ называется *единичной задержкой* или *задержкой на один такт.*

в) Сложение двух чисел, заданных в k-ичной системе счисления.

Рассмотрим функцию $z = x + y$, где k-ичные последовательности x, y складываются по обычным правилам сложения чисел, записанных в k-ичной системе счисления. Так, $z(1) = x(1) + y(1) \pmod k$, $z(2) = x(2)+y(2)+q(2) \pmod k$, где $q(2)$ — «перенос» во второй разряд: $q(2) = 0$, если $x(1) + y(1) < k$, и $q(2) = 1$, если $x(1) + y(1) \geqslant k$, и т.д. Понятно, что разряд $z(t)$ зависит только от разрядов $x(s), y(s)$, где $s = 1, 2, \ldots, t$.

г) По аналогии со сложением рассмотрим функцию возведения в квадрат: $y = x^2$. Используя «школьный» алгоритм умножения, легко убедиться, что для функции $y = x^2$ разряд $y(t)$ зависит только от разрядов $x(1), \ldots, x(t)$.

Далее мы будем рассматривать детерминированные функции только из класса $P_{\partial,k}$. Пусть $f(x_1, \ldots, x_n) \in P_{\partial,k}$ и $\bar a_1 = a_1(1) \ldots a_1(l)$, $\ldots$, $\bar a_n = a_n(1) \ldots a_n(l)$ — произвольные слова длины l в алфавите E_k. Определим *остаточную функцию* $y = f_{\bar a_1 \ldots \bar a_n}(x_1, \ldots, x_n)$, отвечающую словам $\bar a_1, \ldots, \bar a_n$, полагая $y(t)$ равным $(l + t)$-у разряду выходной последовательности функции $f(\bar a_1 x_1, \ldots, \bar a_n x_n)$, где под $\bar a_i x_i$ понимается последовательность $a_i(1) \ldots a_i(l) x_i(1) x_i(2) \ldots$.

Можно сказать, что в последовательности $\bar a_i x_i$ подпоследовательность x_i «задержана» на l тактов, а первые ее l позиций заняты словом $\bar a_i$. Соответственно, для определения выходной последовательности функции $f_{\bar a_1 \ldots \bar a_n}(x_1, \ldots, x_n)$ необходимо из выходной последовательности функции $f(\bar a_1 x_1, \ldots, \bar a_n x_n)$ исключить первые l разрядов. Пользуясь детерминированностью функции f, нетрудно установить, что функция $f_{\bar a_1 \ldots \bar a_n}$ также является детерминированной.

Назовем *весом* детерминированной функции число ее различных остаточных функций. Отметим, что вес функции может быть бесконечным.

Вернемся к примерам а)–г) детерминированных функций. У функции f_φ имеется только одна остаточная функция — она сама. Таким образом, вес функции f_φ равен 1. Функция $з(x)$ имеет k остаточных функций: кроме самой функции $з(x)$ остальные $k - 1$ функций отвечают однобуквенным словам $1, 2, \ldots, k - 1$. Иными словами, если $i \in \{1, 2, \ldots, k - 1\}$,

то соответствующая функция будет получаться из функции з(ix) исключением первого выходного разряда.

В примере в) имеются только две остаточные функции: одна функция исходная (она соответствует случаю, когда в первый разряд перенос отсутствует), другая функция отвечает случаю переноса в первый разряд числа 1. Вторую функцию можно, например, получить, рассматривая сложение вида $(k-1)x + (k-1)y$ (напомним, что младший разряд в последовательности $(k-1)x$ равен $k-1$).

Детерминированная функция из примера г) имеет бесконечный вес. Чтобы в этом убедиться, достаточно, например, рассмотреть бесконечную последовательность слов $\bar{a}$, состоящих только из символов 0.

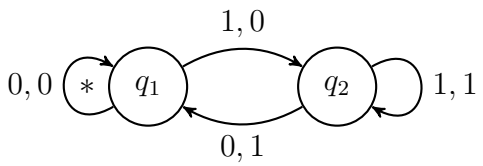
Назовем детерминированную функцию *конечно-автоматной*, если она имеет конечный вес. Множество всех конечно-автоматных функций из $P_{\partial,k}$ обозначим через $P_{\kappa a,k}$.

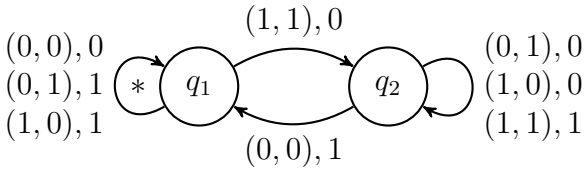
Так же, как и выше, для конечно-автоматных функций применимы способы задания с помощью канонических уравнений и диаграмм Мура. Пусть $f(x_1, \ldots, x_n)$ — конечно-автоматная функция веса r и $f_1, \ldots, f_r$ – все ее остаточные функции (считаем, что $f_1 = f$). Ввиду конечности веса функции f для полного задания функции f достаточно при любом i $(1 \leqslant i \leqslant r)$ и любом наборе $(a_1, \ldots, a_n)$ из E_k^n указать, какая остаточная функция f_j получится из функции f_i, если в качестве первого набора входной последовательности $(x_1, \ldots, x_n)$ будет подан набор $(a_1, \ldots, a_n)$, и какой символ при этом будет на выходе. Поэтому, если остаточные функции $f_1, \ldots, f_r$ считать состояниями $q_1, \ldots, q_r$, то мы сразу приходим к *каноническим уравнениям*, задающим функцию f:

$$\begin{cases} y(t) = F(x_1(t), \ldots, x_n(t), q(t-1)), \\ q(t) = G(x_1(t), \ldots, x_n(t), q(t-1)), \\ q(0) = q_1, \end{cases} \tag{3.3}$$

где функции F и G отображают множество $E_k^n \times \{q_1, \ldots, q_r\}$ в множества E_k и $\{q_1, \ldots, q_r\}$ соответственно. Диаграмму Мура легко построить либо по приведенному выше заданию функции f через ее остаточные функции, либо по каноническим уравнениям. Так же, как в § 2, на дугах диаграммы Мура будут присутствовать две пометки: входной набор $(x_1(t), \ldots, x_n(t))$ из E_k^n и соответствующее ему выходное значение $y(t)$ (также из множества E_k).

В качестве примера приведем диаграммы Мура для функций з(x) и $x + y$ (в обоих случаях $k = 2$):





Чтобы использовать в дальнейшем технику и результаты из теории булевых функций и функций многозначной логики, мы хотим в канонических уравнениях (3.3) придать функциям F, G более «стандартизованную» форму. С этой целью выберем для данного числа r (которое используется в уравнениях (3.3)) число l с условием $k^l \geqslant r$. Закодируем состояния $q_1, \ldots, q_r$ словами длины l в алфавите E_k (при этом в случае неравенства $k^l > r$ некоторые слова длины l окажутся «невостребованными»). Удобно считать, что кодом (начального) состояния q_1 является слово $0 \ldots 0$. Вводя теперь переменные $q_1(t), \ldots, q_l(t)$ для соответствующих разрядов кода, преобразуем уравнения (3.3) в уравнения

$$\begin{cases} y(t) = F(x_1(t), \ldots, x_n(t), q_1(t-1), \ldots, q_l(t-1)), \\ q_1(t) = G_1(x_1(t), \ldots, x_n(t), q_1(t-1), \ldots, q_l(t-1)), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ q_l(t) = G_l(x_1(t), \ldots, x_n(t), q_1(t-1), \ldots, q_l(t-1)), \\ q_1(0) = \ldots = q_l(0) = 0, \end{cases} \tag{3.4}$$

где $F, G_1, \ldots, G_l$ уже являются функциями k-значной логики (в тех случаях, когда переменные $q_1(t-1), \ldots, q_l(t-1)$ под знаками функций $G_1, \ldots, G_l$ дают набор, не являющийся кодом состояния, значения функций $G_1, \ldots, G_l$ можно задать произвольным образом).

Для функций $y = з(x)$ и $y = x_1 + x_2$ (для второй функции $k = 2$) приведем канонические уравнения вида (3.4):

$$\begin{cases} y(t) = q(t-1), \\ q(t) = x(t), \\ q(0) = 0, \end{cases}$$

$$\begin{cases} y(t) = x_1(t) \oplus x_2(t) \oplus q(t-1), \\ q(t) = x_1(t)x_2(t) \vee x_1(t)q(t-1) \vee x_2(t)q(t-1), \\ q(0) = 0. \end{cases}$$

УПРАЖНЕНИЯ

3. Пусть функция $y = f(x)$ принадлежит классу P_2^∞. Выяснить, является ли она детерминированной, если:

1) $y(t) = x(1) \oplus x(2) \oplus \ldots \oplus x(t)$ при $t \geqslant 1$;

2) $y(1) = 1$ и $y(t) = x(t + x(t))$ при $t \geqslant 2$;

3) $y(t) = 0$ при $t \leqslant 50$ и $y(t) = \bar{x}(t - 7)$ при $t > 50$;

4) $f(x) = 0^\infty$, если $x = 0^\infty$, и $f(x) = 1^\infty$ в противном случае (здесь 0^∞ и 1^∞ — соответственно нулевая и единичная последовательности);

5) $f(x) = 1^\infty$, если $x = 0^\infty$, и $y(t) = \bar{x}(t)$ в противном случае ($t \geqslant 1$).

4. Найти все остаточные функции у функции $y = f(x)$, если:

1) $y(1) = 0$ и $y(t) = x(t - 1) \oplus x(t)$ при $t \geqslant 2$;

2) $y(t) = \bar{x}(t)$, если t нечетно, и $y(t) = x(t)$, если t четно;

3) $y(t) = 0$, если $t = i^2$ $(i = 2, 3, \ldots)$, и $y(t) = 1$ в остальных случаях.

5. Выяснить, является ли функция $f \in P_{\partial,2}$ конечно-автоматной и найти ее вес:

1) $y(1) = 1$ и $y(t) = \bar{y}(t - 1)$ при $t \geqslant 2$;

2) $y(t) = x_1(1)$ при $t = 1, 2$ и $y(t) = x_1(t)x_2(t - 1)$ при $t > 2$;

3) $y(t) = \bar{x}_2(t)$, если t нечетно, и $y(t) = x_1(t - 1) \vee x_2(t - 1)$, если t четно;

4) $y(t) = 0$, если число единиц в множестве $\{x_1(1), \ldots, x_n(1), \ldots, x_1(t), \ldots, x_n(t)\}$ четно, и $y(t) = 1$ в противном случае.

§ 4. Операции суперпозиции и введения обратной связи

Так же, как для булевых функций и функций многозначной логики, на множестве $P_{\partial,k}$ можно ввести операцию суперпозиции. Мы будем ее рассматривать лишь в виде

$$f(x_1, \ldots, x_n) = g_0(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)), \tag{3.5}$$

поскольку переход к общему виду трудностей принципиального характера не добавляет.

Теорема 3.3. *Класс $P_{\partial,k}$ замкнут относительно операции суперпозиции.*

Доказательство. Обозначим через $X(t)$ вектор переменных $x_1(t), \ldots, x_n(t)$. Согласно определению детерминированной функции для всякого $t \geqslant 1$ существуют такие функции $G_0^t, G_1^t, \ldots, G_m^t$ из P_k, что для выходных переменных $y_0, y_1, \ldots, y_m$ функций $g_0, g_1, \ldots, g_m$ будем иметь

$$y_0(t) = G_0^t(z_1(1), \ldots, z_m(1), \ldots, z_1(t), \ldots, z_m(t),$$

$$y_i(t) = G_i^t(X(1), \ldots, X(t)) \quad (1 \leqslant i \leqslant m).$$

Тогда, обозначив через y выходную переменную функции f из равенства (3.5), получим

$$y(t) = G_0^t(G_1^1(X(1)), \ldots, G_m^1(X(1)), \ldots, G_1^t(X(1), \ldots, X(t)), \ldots \\ \ldots, G_m^t(X(1), \ldots, X(t))).$$

Отсюда следует детерминированность функции f. Теорема доказана.

Теорема 3.4. *Класс $P_{ка,k}$ замкнут относительно операции суперпозиции.*

Доказательство. Вновь ограничимся суперпозицией вида (3.5). Пусть функции $g_0, g_1, \ldots, g_m$ заданы следующми каноническими уравнениями:

$$\begin{cases} y_0(t) = F_0(y_1(t), \ldots, y_m(t), Q_0(t-1)), \\ Q_0(t) = G_0(y_1(t), \ldots, y_m(t), Q_0(t-1)), \\ Q_0(0) = 0, \end{cases}$$

$$\begin{cases} y_i(t) = F_i(X(t), Q_i(t-1)), \\ Q_i(t) = G_i(X(t), Q_i(t-1)), \\ Q_i(0) = 0, \end{cases}$$

где $X(t)$ есть набор переменных $x_1(t), \ldots, x_n(t)$ и $1 \leqslant i \leqslant m$. Исходя из формулы (3.5), строим канонические уравнения, определяющие функцию f:

$$\begin{cases} y(t) = F_0(F_1(X(t), Q_1(t-1)), \ldots, F_m(X(t), Q_m(t-1)), Q_0(t-1)), \\ Q_0(t) = G_0(F_1(X(t), Q_1(t-1)), \ldots, F_m(X(t), Q_m(t-1)), Q_0(t-1)), \\ Q_1(t) = G_1(X(t), Q_1(t-1)), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ Q_m(t) = G_m(X(t), Q_m(t-1)), \\ Q_0(0) = Q_1(0) = \ldots = Q_m(0) = 0. \end{cases}$$

Теорема доказана.

Из доказательства теоремы 3.4 видно, что если конечно-автоматные функции $g_0, g_1, \ldots, g_m$ имеют соответственно вес $r_0, r_1, \ldots, r_m$, то конечно-автоматная функция f, определяемая равенством (3.5), имеет вес, не превосходящий величины $r_0 \cdot r_1 \cdot \ldots \cdot r_m$. На самом деле эта оценка достижима. Не углубляясь в доказательство достижимости этой оценки в общем случае, укажем лишь на простой пример: l-кратная суперпозиция $з(\ldots з(x)\ldots)$ единичной задержки $з(x)$ имеет вес k^l.

Будем говорить, что детерминированная функция $y = f(x_1, \ldots, x_i, \ldots, x_n)$ зависит от переменной x_i *с запаздыванием*, если при любом t значение $y(t)$ зависит от значений $x_j(1), \ldots, x_j(t)$ при $j \neq i$, от значений $x_i(1), \ldots, x_i(t-1)$ и не зависит существенно от значения $x_i(t)$.

Самым простым и важным примером функции, зависящим от переменной с запаздыванием, является единичная задержка з(x).

Отметим, что в канонических уравнениях для конечно-автоматной функции f, зависящей с запаздыванием от переменной x_i, правая часть уравнения для выходного значения $y(t)$ не зависит существенно от переменной $x_i(t)$.

Перейдем к определению операции *введения обратной связи*. Пусть $\{f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)\}$ — система детерминированных функций, $m \geqslant 2$ и функция f_j зависит с запаздыванием от переменной x_i. Тогда в системе можно ввести обратную связь по переменным x_i и y_j («присоединить» выходную переменную y_j к входной переменной x_i). В результате образуется система (детерминированных) функций $\{f'_1, \ldots, f'_{j-1}, f'_{j+1}, \ldots, f'_m\}$ от переменных $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$.

Формально значения функций f'_l вычисляются следующим образом. Возьмем набор переменных

$$x_1(1), \ldots, x_{i-1}(1), x_{i+1}(1), \ldots, x_n(1). \tag{3.6}$$

Поскольку функция f_j зависит от переменной x_i с запаздыванием, значение $y_j(1)$ можно вычислить на основе набора переменных (3.6). Следовательно, согласно определению обратной связи, при $l \neq j$ значение $y'_l(1)$ будет являться функцией от

$$x_1(1), \ldots, x_{i-1}(1), y_j(1), x_{i+1}(1), \ldots, x_n(1),$$

т.е. от величин (3.6).

Вообще, если значение $y_j(t)$ (в исходной системе функций) вычисляется через значения переменных $x_l(v)$, где $v = 1, \ldots, t$ при $l \neq j$ и $v = 1, \ldots, t-1$ при $l = j$, то после введения обратной связи по переменным x_i и y_j значение $y'_l(t)$ при $l \neq j$ будет являться функцией от

$$x_1(1), \ldots, x_n(1), \ldots, x_1(t-1), \ldots, x_n(t-1), x_1(t), \ldots, x_{i-1}(t), y_j(t), x_{i+1}(t), \ldots, x_n(t),$$

т.е. от значений перечисленных переменных $x_l(v)$, отличных от переменной $x_i(t)$.

Описанный процесс вычисления функций $\{f'_1, \ldots, f'_{j-1}, f'_{j+1}, \ldots, f'_m\}$ показывает, что эти функции также будут являться детерминированными функциями.

Если $f_1, \ldots, f_m$ — конечно-автоматные функции, то операцию введения обратной связи легко определить с использованием канонических уравнений. В самом деле, если выход $y_j(t)$ определяется уравнением

$$y_j(t) = F_j(x_1(t), \ldots, x_{i-1}(t), x_{i+1}(t), \ldots, x_n(t), q_1(t-1), \ldots, q_l(t-1)), \tag{3.7}$$

то правую часть этого уравнения необходимо подставить вместо переменной $x_i(t)$ во все остальные уравнения системы. При этом уравнение (3.7) из системы вычеркивается. Образовавшаяся система канонических уравнений будет определять (конечно-автоматные) функции $\{f'_1, \ldots, f'_{j-1}, f'_{j+1}, \ldots, f'_m\}$, полученные в результате применения операции введения обратной связи по переменным x_i и y_j. Чтобы убедиться в этом, достаточно еще раз просмотреть алгоритм вычисления функций $\{f'_1, \ldots, f'_{j-1}, f'_{j+1}, \ldots, f'_m\}$, изложенный выше для произвольных детерминированных функций.

Как видно из проведенных рассуждений, операция введения обратной связи не приводит к увеличению веса функции.

Полученные выше результаты мы подытожим в виде теоремы.

Теорема 3.5. *Классы $P_{\partial,k}$ и $P_{\kappa a,k}$ замкнуты относительно операции введения обратной связи.*

УПРАЖНЕНИЯ

6. Для суперпозиции $f(x) = f_1(f_2(x))$ построить канонические уравнения и определить вес полученной функции:

$$1)\ f_1: \begin{cases} y_1(t) = \bar{q}_1(t-1), \\ q_1(t) = q_1(t-1) \to x_1(t), \\ q_1(0) = 0, \end{cases} \qquad f_2: \begin{cases} y_2(t) = q_2(t-1), \\ q_2(t) = \bar{x}_2(t), \\ q_2(0) = 0; \end{cases}$$

$$2)\ f_1: \begin{cases} y_1(t) = x_1(t) \cdot q_1(t-1), \\ q_1(t) = \bar{q}_1(t-1), \\ q_1(0) = 0, \end{cases} \qquad f_2: \begin{cases} y_2(1) = 1, \\ y_2(t) = x_2(t) \oplus y_2(t-1),\ t \geqslant 2; \end{cases}$$

$$3)\ f_1: \begin{cases} y_1(1) = 0, \\ y_1(t) = x_1(t-1) \to y_1(t-1), \\ \qquad t \geqslant 2, \end{cases} \qquad f_2: \begin{cases} y_2(t) = q_2(t-1) \to x_2(t), \\ q_2(t) = x_2(t), \\ q_2(0) = 1. \end{cases}$$

7. Построить канонические уравнения и найти вес конечно-автоматной функции, получающейся из функции f введением обратной связи по переменным x_i, y_j:

$$1)\ f:\ \begin{cases} y_1(t) = x_1(t)\cdot q(t-1) \vee x_2(t), \\ y_2(t) = x_2(t), \\ q(t) = (x_1(t) \to x_2(t)) \vee q(t-1), \\ q(0) = 0, \end{cases} \qquad i=1,\ j=2;$$

$$2)\ f:\ \begin{cases} y_1(t) = x_1(t) \oplus q_1(t-1)\cdot q_2(t-1), \\ y_2(t) = x_2(t) \to q_2(t-1), \\ q_1(t) = q_1(t-1) \oplus q_2(t-1), \\ q_2(t) = x_1(t) \oplus q_1(t-1), \\ q_1(0) = q_2(0) = 0, \end{cases}$$

а) $i=1,\ j=2$; б) $i=2,\ j=1$.

§ 5. Конечная порождаемость класса конечно-автоматных функций. Несводимость операции введения обратной связи к операции суперпозиции

В § 3 было определено соответствие между функциями φ k-значной логики и истинностными функциями f_φ из класса $P_{\kappa a,k}$. Нетрудно видеть, что это соответствие сохраняется при операции суперпозиции. В частности, если система функций $\{\varphi_1, \ldots, \varphi_m\}$ полна (относительно суперпозиции) в классе P_k, то система функций $\{f_{\varphi_1}, \ldots, f_{\varphi_m}\}$ будет полной в множестве всех истинностных функций из $P_{\kappa a,k}$.

Мы хотим продолжить исследование в этом направлении и получить конечную полную систему во всем классе $P_{\kappa a,k}$. Однако, как мы увидим позже, одной только операцией суперпозиции в этом случае обойтись невозможно, требуется использовать и операцию введения обратной связи.

Теорема 3.6. *При любом $k \geqslant 2$ класс $P_{\kappa a,k}$ содержит конечную систему функций, полную относительно операций суперпозиции и введения обратной связи.*

Доказательство. Возьмем какую-либо систему функций $\{\varphi_1, \ldots, \varphi_m\}$, полную в классе P_k относительно операции суперпозиции. Пусть $f(x_1, \ldots, x_n)$ — произвольная функция из $P_{\kappa a,k}$, которая определяется

системой канонических уравнений (4). Мы не изменим функцию, определяемую системой уравнений (3.4), если в правых частях уравнений (3.4) вместо функций $F, G_1, \ldots, G_l$ будем рассматривать соответствующие истинностные функции $f_F, f_{G_1}, \ldots, f_{G_l}$. Как отмечено выше, данные функции можно реализовать суперпозициями истинностных функций $\{f_{\varphi_1}, \ldots, f_{\varphi_m}\}$. Из системы уравнений (3.4) образуем систему

$$\begin{cases} y(t) = f_F(x_1(t), \ldots, x_n(t), з(q_1)(t), \ldots, з(q_l)(t)), \\ q_1(t) = f_{G_1}(x_1(t), \ldots, x_n(t), з(q'_1)(t), \ldots, з(q'_l)(t)), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ q_l(t) = f_{G_l}(x_1(t), \ldots, x_n(t), з(q'_1)(t), \ldots, з(q'_l)(t)), \end{cases} \tag{3.8}$$

где наряду с переменными $x_1, \ldots, x_n$ рассматриваются также новые входные переменные $q'_1, \ldots, q'_l$ и «новые» выходные переменные $q_1, \ldots, q_l$. Сравнивая системы (3.4) и (3.8), замечаем, что система (3.8) будет корректно определять функцию f, если переменные $q'_1, \ldots, q'_l$ заменить соответственно переменными $q_1, \ldots, q_l$ (значения переменных $q_1, \ldots, q_l$ при $t = 0$ учитываются в определении единичной задержки з). Вместе с тем, как видно из уравнений системы (3.8), функции $q_1, \ldots, q_l$ зависят с запаздыванием от переменных $q'_1, \ldots, q'_l$. Это значит, что указанную замену переменных можно выполнить, если последовательно вводить обратную связь по парам переменных $(q_1, q'_1), \ldots, (q_l, q'_l)$. Таким образом, по отношению к операциям суперпозиции и введения обратной связи полной в классе $P_{ка,k}$ будет система $\{f_{\varphi_1}, \ldots, f_{\varphi_m}, з\}$. Теорема доказана.

Напомним, что через $V(x_1.x_2)$ мы обозначили функцию Вебба, которая образует полную в P_k систему.

Следствие. *При любом $k \geqslant 2$ система функций $\{f_V, з\}$ полна в классе $P_{ка,k}$ относительно операций суперпозиции и введения обратной связи.*

Теорема 3.7. *При любом $k \geqslant 2$ в классе $P_{ка,k}$ имеется такая функция $f(x_1, x_2, x_3, x_4)$, что система $\{f\}$ полна в классе $P_{ка,k}$ относительно операций суперпозиции и введения обратной связи.*

Доказательство. Рассмотрим следующую функцию F из класса P_k:

$$F(x_1, x_2, x_3, x_4) = \max(x_1 x_4 + x_3(1 - x_4), x_2) + 1 \pmod{k}.$$

Из определения следует, что

$$F(x_1, x_2, x_1, x_4) = \max(x_1, x_2) + 1 = V(x_1, x_2),\ F(1, 0, 0, x_4) = x_4 + 1. \tag{3.9}$$

Положим

$$f(x_1, x_2, x_3, x_4) = f_F(x_1, x_2, x_3, з(x_4)).$$

Соотношения (3.9) показывают, что суперпозициями функции f можно получить функции f_V, f_0, f_1, f_{x+1}. Следовательно, можно также получить функцию $з(x)+1$ и, значит, функцию $з(x)$. Обращение к следствию из теоремы 3.6 завершает доказательство теоремы.

В связи с доказанной теоремой 3.6 возникает вопрос: можно ли в данной теореме отказаться от операции введения обратной связи? Другими словами: нельзя ли операцию введения обратной связи выразить через операцию суперпозиции? Мы покажем, что ответ на эти вопросы отрицательный.

В основе проводимых ниже доказательств лежит следующий нетрудно доказываемый факт: если $f(X)$ — конечно-автоматная функция веса r, а A — периодическая последовательность с длиной периода p, то $f(A)$ — также периодическая последовательность с длиной периода $r_1 p$, где $r_1 \leqslant r$. Поэтому, если имеются конечно-автоматные функции с весами, не превосходящими r, то любая суперпозиция этих функций при применении к периодической последовательности с длиной периода p будет давать периодическую последовательность с длиной периода вида $r_1 \cdot \ldots \cdot r_s \cdot p$, где все числа $r_1, \ldots, r_s$ не превосходят r. Отсюда уже нетрудно вывести существование конечно-автоматной функции, не представимой в виде суперпозиции данных функций.

Как и выше, посредством X обозначаем набор переменных $x_1, \ldots, x_n$.

Теорема 3.8. *Пусть $f(X)$ — конечно-автоматная функция веса r, A — периодическая последовательность с длиной периода p и $B = f(A)$. Тогда последовательность B периодическая с длиной периода $r_1 p$, где $r_1 \leqslant r$.*

Доказательство. Из условия периодичности последовательности A следует, что для некоторого t_0 при всех $t \geqslant t_0$ выполняется равенство

$$A(t + p) = A(t).$$

Возьмем для функции f диаграмму Мура с r вершинами $q_1, \ldots, q_r$. Обозначим через $q(t)$ вершину, которая образуется в диаграмме в момент времени t при подаче на вход последовательности A. Рассмотрим последовательность из $r + 1$ состояний

$$q(t_0),\ q(t_0 + p), \ldots, q(t_0 + rp).$$

Поскольку функция q принимает лишь r значений, эта последовательность содержит повторения. Пусть, например,

$$q(t_0 + ip) = q(t_0 + jp), \quad \text{где} \quad 0 \leqslant i < j \leqslant r.$$

Положим $r_1 = j - i$. Тогда, конечно, $r_1 \leqslant r$. Далее, в силу периодичности последовательности A (с длиной периода p) последовательность

$$q(t_0 + ip), q(t_0 + ip + 1), \ldots, q(t_0 + jp), \ldots$$

будет также периодической с длиной периода $r_1 p$. А тогда и последовательность B будет периодической с той же самой длиной периода. Теорема доказана.

Теорема 3.9. *Пусть $f_0, f_1, \ldots, f_m$ — конечно-автоматные функции с весами $r_0, r_1, \ldots, r_m$, не превосходящими r,*

$$f(X) = f_0(f_1(X), \ldots, f_m(X)),$$

A — периодическая последовательность, длина периода которой содержит лишь простые сомножители, не превосходящие r, и $B = f(A)$. Тогда B — периодическая последовательность, длина периода которой содержит лишь простые сомножители, не превосходящие r.

Доказательство. Положим

$$B_1 = f_1(A), \ \ldots, \ B_m = f_m(A).$$

Согласно теореме 3.8 последовательности $B_1, \ldots, B_m$ периодические, причем для некоторых чисел $r'_1, \ldots, r'_m$, не превосходящих r, длины периодов этих последовательностей имеют вид $r'_1 p, \ldots, r'_m p$, где p — длина периода последовательности A. Понятно, что вектор-последовательность $(B_1, \ldots, B_m)$ также периодическая, причем длина ее периода равна

$$p_0 = \text{НОК}(r'_1 p, \ldots, r'_m p).$$

В частности, все простые сомножители числа p_0 не превосходят r. Положим $C = f_0(B_1, \ldots, B_m)$. Вновь обращаясь к теореме 3.8, видим, что последовательность C периодическая и длина ее периода имеет вид $r'_0 p_0$, где $r'_0 \leqslant r_0$. Теорема доказана.

Теорема 3.10. *Класс $P_{ка,k}$ не содержит конечной системы функций, полной относительно операции суперпозиции.*

Доказательство. Предположим, напротив, что такая система функций $\{f_1, \ldots, f_m\}$ существует, и $r_1, \ldots, r_m$ — веса функций $f_1, \ldots, f_m$. Пусть p — простое число и $p > \max(r_1, \ldots, r_m)$. Определим конечно-автоматную функцию $f(x)$, которая перерабатывает любую входную последовательность в периодическую последовательность с периодом $0^{p-1}1$. Используя теорему 3.9, получаем, что функцию f невозможно реализовать суперпозициями функций $f_1, \ldots, f_m$. Теорема доказана.

УПРАЖНЕНИЯ

8. Используя полную в классе $P_{\mathcal{K}a,2}$ систему конечно-автоматных функций $\{f_\vee, f_\&, f_-, \text{з}\}$, определить с помощью операций суперпозиции и введения обратной связи следующие конечно-автоматные функции:

1) $\begin{cases} y(t) = x(t) \oplus q(t-1), \\ q(t) = \bar{x}(t) \cdot \bar{q}(t-1), \\ q(0) = 0, \end{cases}$ 2) $\begin{cases} y_1(t) = \bar{x}_1(t) \vee x_2(t) \vee q_1(t-1), \\ y_2(t) = \bar{q}_1(t-1) \vee \bar{x}_2(t) \cdot q_2(t-1), \\ q_1(t) = x_2(t) \to q_1(t-1), \\ q_2(t) = \bar{q}_2(t-1) \vee x_1(t), \\ q_1(0) = q_2(0) = 0. \end{cases}$

9. Доказать полноту системы функций $\{f_1, f_2\}$ в классе $P_{\mathcal{K}a,2}$ относительно операций суперпозиции и введения обратной связи:

1) $f_1 :\ y(t) = \bar{x}_1(t) \cdot \bar{x}_2(t)\ (t \geqslant 1)$,

$$f_2 : \begin{cases} y(t) = x_1(t) \cdot \bar{x}_2(t) \vee q(t-1), \\ q(t) = x_1(t) \vee x_2(t), \\ q(0) = 0; \end{cases}$$

2) $f_1 :\ y(t) = x_1(t) \to \bar{x}_2(t)\ (t \geqslant 1)$,

$$f_2 : \begin{cases} y(t) = (x_1(t) \to x_2(t)) \cdot q(t-1), \\ q(t) = x_1(t) \cdot x_2(t), \\ q(0) = 0. \end{cases}$$

Глава 4

МАШИНЫ ТЬЮРИНГА И ВЫЧИСЛИМЫЕ ФУНКЦИИ

§ 1. Машина Тьюринга

В середине 1930-х годов нескольким математикам (А. Чёрч, К. Гёдель, А. Тьюринг, Э. Пост, С. Клини) и с использованием различных подходов удалось получить точные математические формулировки для понятий алгоритма и алгоритмически вычислимой функции. Один из этих подходов был предложен английским математиком А.Тьюрингом (A.Turing, 1912–1954) и базировался на некотором абстрактном вычислительном устройстве, которое впоследствии было названо *машиной Тьюринга*. Чуть позже почти такое же понятие абстрактной вычислительной машины было предложено американским математиком Э. Постом (E. Post, 1897–1954). Поэтому иногда машины Тьюринга называют машинами Тьюринга-Поста. Мы будем использовать более распространенное название *машина Тьюринга*.

Машина Тьюринга состоит из *ленты*, *считывающе-записывающей головки* и *управляющего устройства* (см. рис. 1).

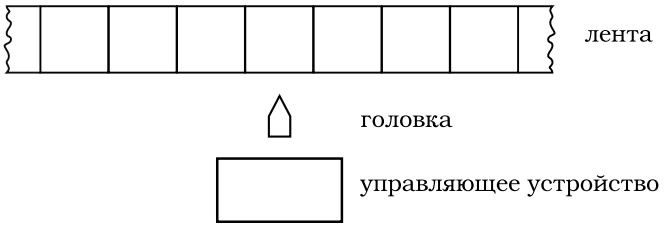


Рис. 1: Машина Тьюринга

Лента бесконечна влево и вправо и разбита на клетки. В каждой клетке ленты может быть записан один из символов (букв) конечного ленточного алфавита $A = \{a_0, a_1, \ldots, a_k\}$. Обычно в алфавите A выделяется так называемый «пустой» символ a_0.

Головка машины может:

двигаться по ленте влево и вправо, перемещаясь из одной клетки в соседнюю с ней клетку;

читать символ, записанный в клетке, и записывать в обозреваемую клетку любой символ алфавита A.

Управляющее устройство организует перемещение головки на ленте и запись символов в клетки ленты. Оно может находиться в одном из конечного числа *состояний* $q_0, q_1, \ldots, q_r$. В множестве состояний выделяются *начальное состояние* q_1 и *заключительное состояние* q_0.

Изменение положения головки на ленте, запись новых символов в клетки ленты и переход в другие состояния происходит согласно *программе* машины, которая состоит из *команд* вида

$$a_i q_j \to a_l D q_s, \tag{4.1}$$

где $j \neq 0$ и D есть один из символов движения: L, R, S. Для любых возможных значений i, j $(0 \leqslant i \leqslant k,\ 1 \leqslant j \leqslant r)$ программа машины содержит ровно одну команду (4.1) с левой частью $a_i q_j$. Обычно программу машины Тьюринга записывают в виде таблицы, строки которой помечены символами $a_0, a_1, \ldots, a_k$, а столбцы — символами $q_1, \ldots, q_r$. В клетке таблицы, отвечающей строке a_i и столбцу q_j, записана правая часть $a_l D q_s$ команды (4.1).

Работа машины Тьюринга протекает в дискретном времени $t = 1, 2, \ldots$. Перед началом работы машины Тьюринга $(t = 0)$ на ленте записывается некоторое слово $\bar{a}$ в алфавите $\{a_1, \ldots, a_k\}$, вся остальная часть ленты (слева и справа от слова $\bar{a}$) заполняется «пустым» символом a_0. Головка машины Тьюринга устанавливается в клетку, содержащую первую букву слова $\bar{a}$, а управляющее устройство приводится в состояние q_1.

Каждый из последующих незаключительных шагов (тактов) работы машины Тьюринга выполняется согласно одному и тому же правилу: если в момент времени t головка машины Тьюринга считывает символ a_i, управляющее устройство находится в состоянии q_j $(j \neq 0)$ и в программе машины имеется команда (4.1), то в момент времени $t + 1$:

в обозреваемую в момент t клетку ленты будет записан символ a_l (возможно, что $l = i$);

головка сдвинется на одну клетку влево (если $D = L$), вправо (если $D = R$) или останется в прежней клетке (если $D = S$);

управляющее устройство перейдет в состояние q_s (и здесь возможно равенство $s = j$).

Машина Тьюринга заканчивает работу, если управляющее устройство попадает в заключительное состояние q_0.

Сразу отметим, что, вообще говоря, не для всех слов $\bar{a}$ машина Тьюринга завершает свою работу. Для некоторых слов $\bar{a}$ она может работать неограниченно долго, не попадая в заключительное состояние q_0. В слу-

чае окончания работы машины Тьюринга результатом применения машины к слову $\bar{a}$ обычно считают слово в алфавите $\{a_1, \ldots, a_k\}$, которое оказывается записанным на ленте в заключительный момент времени. Если же машина Тьюринга не заканчивает работу, то результат применения машины к слову $\bar{a}$ не определен.

Рассмотрим примеры машин Тьюринга. Машина Тьюринга $\mathcal{M}_1$, начиная работу на первом символе a_1 слова $\bar{a} = a_1 \ldots a_1$, оставляет его без изменения, сдвигает головку на одну клетку вправо и останавливается.

Машина Тьюринга $\mathcal{M}_2$ также оставляет первый символ a_1 слова $\bar{a} = a_1 \ldots a_1$ без изменения, однако сдвигает головку на одну клетку влево, записывает в нее еще один символ a_1 и останавливается. Машина Тьюринга $\mathcal{M}_3$ на любом слове работает неограниченно долго.

	q_1
a_0	$a_0 R q_1$
a_1	$a_1 R q_0$

$\mathcal{M}_1$

	q_1
a_0	$a_1 S q_0$
a_1	$a_1 L q_1$

$\mathcal{M}_2$

	q_1
a_0	$a_0 S q_1$
a_1	$a_1 S q_1$

$\mathcal{M}_3$

В дальнейшем нас в первую очередь будут интересовать функции, вычислимые на машинах Тьюринга. Чтобы определить эти функции, необходимо прежде всего условиться о способе представления чисел из множества $N = \{0, 1, 2, \ldots\}$ (все вычислимые функции расматриваются, как правило, на множестве N) на ленте машины Тьюринга. Мы выберем способ, который на практике встречается редко, однако удобен в теоретических построениях. Именно, число m из N будем представлять словом, состоящем из $m + 1$ символов 1 («лишнюю» единицу добавляем, чтобы иметь возможность представлять число 0). В связи с этим ленточный алфавит A машины Тьюринга, предназначенной для вычисления некоторой функции, всегда будет содержать символ 1 (например, $a_1 = 1$), пустой символ 0 ($a_0 = 0$) и, возможно, другие символы.

Итак, число m из N записываем на ленте машины Тьюринга в виде массива из $m+1$ единиц, а в остальных клетках ленты помещаем пустой символ 0. Если $n > 1$ и требуется представить на ленте машины Тьюринга набор чисел $(m_1, \ldots, m_n)$, то образуем n массивов из $m_1+1, \ldots, m_n+1$ единиц соответственно и помещаем между соседними массивами разделительный символ 0. Такое представление набора чисел $(m_1, \ldots, m_n)$ на ленте машины Тьюринга (включая случай $n = 1$) будем называть *основным кодом набора* $(m_1, \ldots, m_n)$.

Перейдем собственно к определению функции, вычислимой на машине Тьюринга. Пусть $f(x_1, \ldots, x_n)$ — функция (вообще говоря, частичная, т.е. определенная не для всех значений аргументов) с аргументами, принимающими значения из N, и значениями также в множестве N, $\mathcal{M}$ — машина Тьюринга с ленточным алфавитом A, содержащим символы 0 и 1. Будем говорить, что *машина $\mathcal{M}$ вычисляет функцию f*, если для любого набора $(m_1, \ldots, m_n)$ выполняются следующие условия.

1. Если значение $f(m_1, \ldots, m_n)$ определено, то машина $\mathcal{M}$, начиная работу в состоянии q_1 на первой единице основного кода набора $(m_1, \ldots, m_n)$, через конечное число тактов останавливается и в заключительный момент времени на ленте машины в основном коде представлено число $f(m_1, \ldots, m_n)$.

2. Если значение $f(m_1, \ldots, m_n)$ не определено, то, начиная работу из той же позиции, что и в п. 1, машина $\mathcal{M}$ либо никогда не останавливается, либо останавливается, но при этом на ленте машины не представлено в основом коде никакое число из N.

В дальнейшем по чисто техническим причинам нам будет удобно иметь определение вычислимой функции в более стандартизованном виде. Именно, мы хотим, чтобы в п. 1 определения машина $\mathcal{M}$ в заключительный момент времени останавливалась на первой единице основного кода числа $f(m_1, \ldots, m_n)$. А в п. 2 определения мы хотим отказаться от второй возможности, когда на ленте машины $\mathcal{M}$ отсутствует основной код числа из N. Такое модифицированное понятие вычисления функции f на машине $\mathcal{M}$ будем называть *правильным вычислением функции f на машине $\mathcal{M}$*. Сравнительно несложно показать, что второе определение вычислимости не сужает класса функций, вычислимых на машинах Тьюринга.

Обратимся к примерам вычисления функций на машинах Тьюринга.

	q_1
0	$1Sq_0$
1	$1Lq_1$

$\mathcal{M}_4$

	q_1
0	$1Sq_0$
1	$0Rq_1$

$\mathcal{M}_5$

	q_1	q_2
0	—	$1Sq_0$
1	$0Rq_2$	$1Sq_0$

$\mathcal{M}_6$

Нетрудно понять, что машина $\mathcal{M}_4$ правильно вычисляет функцию $x+1$. Также нетрудно убедиться, что машина $\mathcal{M}_5$ правильно вычисляет функцию-константу 0 (рассматриваемую от одной переменной). Машина $\mathcal{M}_5$ пробегает в состоянии q_1 слева направо весь массив из единиц и

заменяет их нулями. После этого вместо первого справа символа 0 она ставит 1 и останавливается.

Рассмотрим чуть более сложный пример функции

$$x \dot{-} 1 = \begin{cases} 0, & \text{если } x = 0, \\ x - 1 & \text{в остальных случаях.} \end{cases}$$

Машина Тьюринга $\mathcal{M}_6$ заменяет первую единицу основного кода нулем, переходит в состояние q_2 и сдвигает головку вправо. Находясь в состоянии q_2, она записывает во вторую клетку 1 и останавливается. Прочерк в одной из клеток таблицы для машины $\mathcal{M}_6$ указывает на то, что эту клетку можно заполнить произвольным образом.

Из простых вычислимых функций остановимся еще на *селекторных функциях* $I_i^n(x_1, \ldots, x_i, \ldots, x_n) = x_i$. Все они вычисляются сходным образом: головка машины движется слева направо до i-го массива из единиц, стирая по пути все единицы (т.е. заменяя их нулями), оставляет без изменения i-й массив, затем, продвигаясь до конца основного кода, стирает единицы оставшихся массивов. В заключение головка машины возвращается на первую единицу бывшего i-го массива. Поскольку число состояний машины Тьюринга, вычисляющей функцию I_i^n, линейным образом зависит от параметров i, n, мы построим машину Тьюринга лишь для вычисления функции $I_2^3(x_1, x_2, x_3)$. Программа этой машины представлена в нижеследующей таблице.

	q_1	q_2	q_3	q_4	q_5
0	$0Rq_2$	$0Rq_3$	$0Lq_4$	$0Lq_4$	$0Rq_0$
1	$0Rq_1$	$1Rq_2$	$0Rq_3$	$1Lq_5$	$1Lq_5$

В заключение параграфа сформулируем *тезис Тьюринга.*
Всякая алгоритмически вычислимая функция может быть вычислена на подходящей машине Тьюринга.

УПРАЖНЕНИЯ

1. Доказать, что для всякой машины Тьюринга существует эквивалентная ей (т.е. дающая тот же самый результат) машина Тьюринга, программа которой не содержит команд с символом S.

2. Построить машину Тьюринга, правильно вычисляющую функцию $x + y$.

3. Может ли одна и та же машина Тьюринга правильно вычислять две различные функции?

4. Пусть программа машины Тьюринга $\mathcal{M}'$ получена из программы машины $\mathcal{M}$ заменой всех символов L символами R и всех символов R символами L. Можно ли в каком-либо смысле утверждать, что машина $\mathcal{M}'$ вычисляет те же функции, что и машина $\mathcal{M}$?

§ 2. Композиция и итерация машин Тьюринга

Обычно при построении достаточно сложных машин Тьюринга прибегают к известному на практике приему: разбивают алгоритм, реализуемый машиной Тьюринга, на отдельные простые части, строят для них машины Тьюринга и затем «собирают» из полученных «малых» машин Тьюринга требуемую машину Тьюринга. Если разбиение алгоритма на части и построение для них машин Тьюринга есть процесс в значительной степени творческий, то последующая «сборка» машин Тьюринга является делом весьма рутинным. Здесь существуют два основных приема конструирования из одних машин Тьюринга других машин Тьюринга. Это *композиция* и *итерация* машин Тьюринга.

Композиция машин Тьюринга формализует идею последовательного выполнения двух (реже нескольких) вычислительных процессов. В самом простейшем случае композиция двух машин Тьюринга определяется так. Пусть $\mathcal{M}_1, \mathcal{M}_2$ — машины Тьюринга с одним и тем же ленточным алфавитом. Будем предполагать, что множества состояний машин $\mathcal{M}_1, \mathcal{M}_2$ не пересекаются. Мы хотим создать машину Тьюринга $\mathcal{M}$, которая работает следующим образом. На исходном слове $\bar{a}$ начинает работать машина $\mathcal{M}_1$. Если $\mathcal{M}_1$ заканчивает свою работу, то с этой конфигурации (т.е. с достигнутыми заполнением ленты и положением головки на ленте) запускается машина $\mathcal{M}_2$. Результат, полученный при этом машиной $\mathcal{M}_2$, и будет являться результатом применения машины $\mathcal{M}$ к слову $\bar{a}$. Разумеется, если одна из машин $\mathcal{M}_1, \mathcal{M}_2$ работает неограниченно долго, то и машина $\mathcal{M}$ будет работать неограниченно долго.

Довольно понятно, как из программ машин $\mathcal{M}_1, \mathcal{M}_2$ получить программу машины $\mathcal{M}$. Для этого необходимо во всех командах машины $\mathcal{M}_1$ заменить заключительное состояние начальным состоянием машины $\mathcal{M}_2$, а затем объединить полученные множества команд машин $\mathcal{M}_1$ и $\mathcal{M}_2$. Начальным состоянием машины $\mathcal{M}$ будет являться начальное состояние машины $\mathcal{M}_1$, а заключительным — заключительное состояние машины $\mathcal{M}_2$.

Вернемся к машине Тьюринга из предыдущего параграфа, которая правильно вычисляет функцию $I_2^3(x_1, x_2, x_3)$. Ее можно получить в виде последовательной композиции машин Тьюринга $\mathcal{M}_1$ – $\mathcal{M}_5$ (первый индекс у состояния указывает на номер машины):

	q_{11}
0	$0Rq_{10}$
1	$0Rq_{11}$

$\mathcal{M}_1$

	q_{21}
0	$0Rq_{20}$
1	$1Rq_{21}$

$\mathcal{M}_2$

	q_{31}
0	$0Lq_{30}$
1	$0Rq_{31}$

$\mathcal{M}_3$

	q_{41}
0	$0Lq_{41}$
1	$1Lq_{40}$

$\mathcal{M}_4$

	q_{51}
0	$0Rq_{50}$
1	$1Lq_{51}$

$\mathcal{M}_5$

Машина $\mathcal{M}_1$ стирает в основном коде набора (x_1, x_2, x_3) первый массив из единиц, пропускает пустую клетку между первым и вторым массивами и останавливается на первой единице второго массива. Машина $\mathcal{M}_2$, ничего не изменяя, пробегает второй массив из единиц, пропускает следующую за ним пустую клетку и останавливается на первой единице третьего массива. Действия машины $\mathcal{M}_3$ аналогичны действиям машины $\mathcal{M}_1$, за исключением последнего шага, на котором головка машины $\mathcal{M}_3$ смещается на одну клетку влево. Машина $\mathcal{M}_4$ движется влево по пустым клеткам до крайней правой единицы бывшего второго массива и останавливается в соседней с ней клетке. Наконец, машина $\mathcal{M}_5$ пробегает справа налево оставшийся массив из единиц, выходит на примыкающую к нему пустую клетку, возвращается на одну клетку назад и останавливается.

Тот тип композиции, который мы описали выше, можно было бы назвать безусловной композицией машин Тьюринга. Однако существуют задачи, где необходимо воспользоваться более сложными формами композиции. Предположим, например, что мы хотим получить из данных машин Тьюринга $\mathcal{M}_1$, $\mathcal{M}_2$ такую машину Тьюринга $\mathcal{M}$, которая в некоторых случаях работает как машина $\mathcal{M}_1$, а в других — как композиция машин $\mathcal{M}_1$ и $\mathcal{M}_2$. Для этого, исходя из содержательных соображений, мы выделим среди заключительных команд (т.е. команд, содержащих заключительное состояние) машины $\mathcal{M}_1$ такие команды, которые отвечают второй из рассматриваемых возможностей. А далее, как и выше, заменим в них заключительное состояние начальным состоянием машины $\mathcal{M}_2$. Все остальные команды обеих машин оставим без изменения.

Еще один тип композиции машин Тьюринга соответствует случаю, когда необходимо осуществить выбор из двух или более альтернатив. Пусть $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ — машины Тьюринга с попарно не пересекающимися множествами состояний. И пусть содержательно машина $\mathcal{M}_1$ распознает

некоторое свойство (предикат) p. При этом в случае выполнения свойства p мы бы хотели запустить машину $\mathcal{M}_2$, а в случае невыполнения — машину $\mathcal{M}_3$. Формально свойство p характеризуется заключительными командами машины $\mathcal{M}_1$: часть заключительных команд соответствует выполнению свойства p, остальная часть — невыполнению свойства p. Обозначим заключительное состояние из первой части заключительных команд машины $\mathcal{M}_1$ через q_0', из второй части — через q_0''. Тогда указанное «разветвление» машин Тьюринга $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ выполняется так: все заключительные состояния q_0' машины $\mathcal{M}_1$ заменяются начальным состоянием машины $\mathcal{M}_2$, а все заключительные состояния q_0'' — начальным состоянием машины $\mathcal{M}_3$. Заключительными состояними полученной машины Тьюринга являются заключительное состояние машины $\mathcal{M}_2$ и заключительное состояние машины $\mathcal{M}_3$.

Рассмотрим пример композиции машин Тьюринга последнего типа. Используя этот тип композиции, построим машину Тьюринга, которая правильно вычисляет функцию

$$\mathrm{rm}(x,2) = \begin{cases} 0, & \text{если } x \text{ четно}, \\ 1, & \text{если } x \text{ нечетно}. \end{cases}$$

Определим машины $\mathcal{M}_6$, $\mathcal{M}_7$, $\mathcal{M}_8$.

	q_{11}	q_{12}
0	$0Sq_{10}''$	$0Sq_{10}'$
1	$0Rq_{12}$	$0Rq_{11}$

$\mathcal{M}_6$

	q_{21}
0	$1Sq_{20}$
1	—

$\mathcal{M}_7$

	q_{31}	q_{32}
0	$1Rq_{32}$	$1Lq_{30}$
1	—	—

$\mathcal{M}_8$

Машина $\mathcal{M}_6$, находясь в начальном состоянии q_{11} на первой единице массива из $x+1$ единиц, движется по массиву вправо, стирает все единицы и поочередно проходит через состояния q_{11}, q_{12}. В случае четности числа x (тогда пробегаемый массив состоит из нечетного числа единиц) машина $\mathcal{M}_6$ выходит на первую пустую клетку в состоянии q_{12} и останавливается в заключительном состоянии q_{10}'. В случае нечетности числа x аналогичный процесс заканчивается в заключительном состоянии q_{10}''. Таким образом, с помощью состояний q_{10}', q_{10}'' машина $\mathcal{M}_6$ «распознает» четность числа x.

Машина $\mathcal{M}_7$ записывает в пустую клетку 1 (основной код числа 0) и останавливается. Машина $\mathcal{M}_8$ записывает в двух пустых клетках единицы, возвращается на первую единицу и останавливается. Прочерки в

таблицах для машин $\mathcal{M}_7$, $\mathcal{M}_8$ означают, что соответствующие клетки могут быть заполнены произвольным образом.

Образуем теперь композицию $\mathcal{M}$ машин $\mathcal{M}_6, \mathcal{M}_7, \mathcal{M}_8$ так, чтобы машина $\mathcal{M}_7$ начинала работу при попадании машины $\mathcal{M}_6$ в заключительное состояние q'_{10}, а машина $\mathcal{M}_8$ — в заключительное состояние q''_{10}. Соответствующая таблица для машины $\mathcal{M}$ будет выглядеть следующим образом:

	q_{11}	q_{12}	q_{21}	q_{31}	q_{32}
0	$0Sq_{31}$	$0Sq_{21}$	$1Sq_{20}$	$1Rq_{32}$	$1Lq_{30}$
1	$0Rq_{12}$	$0Rq_{11}$	—	—	—

Итерация машины Тьюринга формализует идею многократного регулируемого применения одного и того же алгоритма. Пусть задана машина Тьюринга $\mathcal{M}$ и мы хотим многократно применять эту машину (в духе композиции машин Тьюринга) до тех пор, пока не будет выполнено некоторое условие (предикат) p. Само условие p, как и ранее, можно задать с помощью некоторых заключительных команд машины $\mathcal{M}$. Чтобы выделить эти заключительные команды, обозначим содержащееся в них заключительное состояние через q'_0. Все остальные вхождения заключительного состояния в программу машины $\mathcal{M}$ обозначим через q''_0. Теперь наше намерение итерировать работу машины Тьюринга $\mathcal{M}$ можно выразить более точно. Запускаем машину $\mathcal{M}$ в начальном состоянии q_1. Если через некоторое количество тактов она попадает в заключительное состояние q''_0, то «возвращаем» ее вновь в начальное состояние q_1 и т.д. Формально итерация машины $\mathcal{M}$ относительно условия p получается заменой всех вхождений заключительного состояния q''_0 начальным состоянием q_1.

Пусть, например, машина $\mathcal{M}$ задается таблицей 1.

	q_1	q_2	q_3	q_4
0	—	$0Rq_2$	$0Sq'_0$	$0Lq_4$
1	$0Rq_2$	$0Rq_3$	$0Lq_4$	$1Sq''_0$

Табл. 1

	q_1	q_2	q_3	q_4
0	—	$0Rq_2$	$0Sq'_0$	$0Lq_4$
1	$0Rq_2$	$0Rq_3$	$0Lq_4$	$1Sq_1$

Табл. 2

Проанализируем работу машины $\mathcal{M}$. Предположим, что в начальный момент времени на ленте машины $\mathcal{M}$ записан основной код набора (x_1, x_2), а головка машины установлена на крайнюю правую единицу первого массива. Начиная работу в этой позиции, машина $\mathcal{M}$ совершает следующие действия.

Она заменяет обозреваемую единицу нулем, сдвигается вправо до первой единицы второго массива и также заменяет ее нулем. Если $x_2 = 0$, то машина $\mathcal{M}$ останавливается в следущей справа клетке в состоянии q_0'. В противном случае машина $\mathcal{M}$ заменяет вторую единицу второго массива нулем и возвращается (если $x_1 \neq 0$) на крайнюю правую единицу первого массива, где и останавливается в состоянии q_0''.

Образуем итерацию машины $\mathcal{M}$ — заменим в ее программе заключительное состояние q_0'' начальным состоянием q_1 (см. табл. 2). Получим машину Тьюринга $\mathcal{M}'$. Нетрудно понять, как будет работать машина $\mathcal{M}'$. Она достигает заключительное состояние q_0' в том и только том случае, когда $x_2 = 2y$ и $x_1 \geqslant y$. В остальных случаях машина $\mathcal{M}'$ работает неограниченно долго.

УПРАЖНЕНИЯ

5. Представить в виде композиции более простых машин Тьюринга машины, правильно вычисляющие функции

$$\mathrm{sg}(x) = \begin{cases} 0, & \text{если } x = 0, \\ 1, & \text{если } x > 0, \end{cases} \qquad \overline{\mathrm{sg}}(x) = \begin{cases} 1, & \text{если } x = 0, \\ 0, & \text{если } x > 0, \end{cases}$$

$$3 \dot{-} x = \begin{cases} 0, & \text{если } x \leqslant 3, \\ x - 3, & \text{если } x > 3, \end{cases}$$

6. Применив итерацию машины Тьюринга, построить машину, правильно вычисляющую функцию $[x/2]$ (целая часть от деления x на 2).

§ 3. Моделирование машин Тьюринга

В теории алгоритмов довольно распространенным является прием, когда с помощью одного вычислительного устройства моделируется работа другого вычислительного устройства. Здесь мы рассмотрим лишь моделирвание машин Тьюринга, работающих в алфавите $\{0, 1, a_2, \ldots, a_k\}$, где $k \geqslant 2$, машинами Тьюринга, имеющими ленточный алфавит $\{0, 1\}$. Более того, нас будут интересовать только функции, правильно вычислимые теми и другими машинами Тьюринга. Мы установим, что эти классы функций совпадают.

Идея моделирования машины Тьюринга, работающей в алфавите $A = \{0, 1, a_2, \ldots, a_k\}$, чрезвычайно проста. Необходимо закодировать буквы

алфавита A словами в алфавите $\{0,1\}$ и затем оперировать с этими кодами так же, как исходная машина Тьюринга оперирует с буквами $0, 1, a_2, \ldots, a_k$.

Из большого количество разнообразных кодирований мы выберем блочное (равномерное) кодирование, когда каждой букве алфавита A сопоставляется двоичное слово (блок) одной и той же длины l. Понятно, что для однозначности последующего декодирования число l должно удовлетворять неравенству $2^l \geqslant k+1$. Теперь каждой букве a алфавита A можно сопоставить двоичное слово $\nu(a)$ длины l, причем разным буквам a — различные слова $\nu(a)$. В дальнейшем нам будет удобно считать, что кодирование ν ставит в соответствие букве 0 слово из l нулей, а букве 1 — слово из l единиц.

Пусть $\mathcal{M}$ — произвольная машина Тьюринга с ленточным алфавитом A и множеством состояний $\{q_0, q_1, \ldots, q_r\}$. Мы хотим сначала выполнить центральную часть моделирования работы машины $\mathcal{M}$ — построить машину Тьюринга $\mathcal{M}_1$ с ленточным алфавитом $\{0,1\}$, которая работает над кодами слов так же, как машина $\mathcal{M}$ работает над исходными словами в алфавите A. С этой целью к состояниям $q_0, q_1, \ldots, q_r$ машины $\mathcal{M}$ добавим еще три группы состояний.

Состояния первой группы будем для наглядности записывать в виде $[b_1 \ldots b_p, j]$, где $1 \leqslant p \leqslant l$, $1 \leqslant j \leqslant r$ и $b_1, \ldots, b_p$ — символы 0 или 1. В состояниях первой группы машина $\mathcal{M}_1$ проводит «расшифровку» кодов $\nu(a)$, «помня» при этом текущее состояние машины $\mathcal{M}$.

Предположим, что в некоторый момент времени машина $\mathcal{M}_1$, моделируя работу машины $\mathcal{M}$, находится в состоянии q_j (так же, как и машина $\mathcal{M}$), а ее головка обозревает первую букву кода некоторой буквы алфавита A. Тогда следующие команды в программе машины $\mathcal{M}_1$ обеспечивают чтение этого кода:

$$bq_j \to bR[b, j], \quad b[b_1 \ldots b_p, j] \to bR[b_1 \ldots b_p b, j] \quad (1 \leqslant p \leqslant l-2),$$

$$b[b_1 \ldots b_{l-1}, j] \to bS[b_1 \ldots b_{l-1} b, j]$$

(здесь b, b_i — символы 0 или 1).

Пусть теперь в программе машины $\mathcal{M}$ имеется команда (4.1), $\nu(a_i) = b_1 \ldots b_l$, $\nu(a_r) = c_1 \ldots c_l$ и машина $\mathcal{M}_1$ в некоторый момент времени находится на последней букве b_l кода $\nu(a_i)$ в состоянии $[b_1 \ldots b_l, j]$. Состояния второй группы, которые мы обозначим как $[b_1 \ldots b_p, i, j]$, предназначены для моделирования «половины» выполнения машиной $\mathcal{M}$ команды (4.1),

которое состоит в замене кода $b_1 \dots b_l$ кодом $c_1 \dots c_l$. С этими состояниями связаны следующие команды программы машины $\mathcal{M}_1$:

$$b_l[b_1 \dots b_l, j] \to c_l L[b_1 \dots b_{l-1}, i, j],$$
$$b_p[b_1 \dots b_p, i, j] \to c_p L[b_1 \dots b_{p-1}, i, j] \quad (2 \leqslant p \leqslant l-1),$$
$$b_1[b_1, i, j] \to c_1 S\{D, s\}$$

(здесь символы D, s взяты из правой части команды (4.1), а $\{D, s\}$ есть обозначение нового состояния, которое мы отнесем к третьей группе).

Теперь, находясь в состоянии $\{D, s\}$, машина $\mathcal{M}_1$ должна промоделировать выполнение оставшейся части команды (4.1), т.е. $\mathcal{M}_1$ должна сдвинуться на l клеток влево (если $D = L$) или вправо (если $D = R$) и перейти в состояние q_s. В случае $D = S$ машина $\mathcal{M}_1$, не сдвигая головку, сразу переходит в состояние q_s. Для выполнения этих действий служат состояния третьей группы и соответствующие команды:

$$b\{L, s\} \to bL\{L, s, 1\},\ b\{L, s, p\} \to bL\{L, s, p+1\} \quad (1 \leqslant p \leqslant l-1),$$
$$b\{L, s, l\} \to bSq_s;$$
$$b\{R, s\} \to bR\{R, s, 1\},\ b\{R, s, p\} \to bR\{R, s, p+1\} \quad (1 \leqslant p \leqslant l-1),$$
$$b\{R, s, l\} \to bRq_s;$$
$$b\{S, s\} \to bSq_s.$$

Чтобы полностью промоделировать работу машины $\mathcal{M}$, необходимо определить еще две машины Тьюринга $\mathcal{M}_0$ и $\mathcal{M}_2$. Машина $\mathcal{M}_0$ «растягивает» основной код исходного набора в l раз, а машина $\mathcal{M}_2$, наоборот, «сжимает» основной код числа в l раз. Поскольку действия машин $\mathcal{M}_0,\ \mathcal{M}_2$ в известном смысле обратны друг другу, мы рассмотрим лишь машину $\mathcal{M}_0$.

Итак, машина $\mathcal{M}_0$ должна «растянуть» основной код любого набора в l раз. Это значит, что каждая единица основного кода должна быть заменена l единицами, а каждый нуль, стоящий между массивами из единиц, — l нулями. Для простоты ограничимся случаем, когда машина $\mathcal{M}$ вычисляет функцию от одной переменной.

Машину Тьюринга $\mathcal{M}_0$ можно представить в виде композиции и итерации следующих машин $\mathcal{M}_3$ – $\mathcal{M}_6$. Машина $\mathcal{M}_3$, находясь на первой единице основного кода числа x, сдвигается вправо на одну клетку и анализирует содержащийся в ней символ a. Если $a = 0$, то запускается машина $\mathcal{M}_4$, которая дописывает к имеющейся на ленте единице еще $l-1$ единиц и останавливается. Если $a = 1$, то запускается машина $\mathcal{M}_5$, которая стирает первую единицу основного массива, сдвигается влево на

одну клетку и записывает массив из l единиц. Далее действует машина $\mathcal{M}_6$, которая будет подвергнута итерации.

Машина $\mathcal{M}_6$ пробегает слева направо массив из единиц, затем массив из нулей, встречает первую единицу следующего массива и анализирует символ a, стоящий непосредственно справа от этой единицы. Если $a = 1$, то стирается первая единица второго массива, машина $\mathcal{M}_6$ движется далее налево, пробегает массив из нулей, затем массив из единиц и приписывает к последнему массиву слева новые l единиц. После чего $\mathcal{M}_6$ переходит в начальное состояние (цикл итерации). Если же $a = 0$, то машина $\mathcal{M}_6$ действует аналогично до момента перехода в начальное состояние. После чего машина $\mathcal{M}_6$ должна завершить работу в заключительном состоянии.

Построение машин $\mathcal{M}_3$ – $\mathcal{M}_6$ не представляет большой трудности, и мы его опускаем.

УПРАЖНЕНИЯ

7. Построить программу машины Тьюринга $\mathcal{M}_6$.

§ 4. Операции суперпозиции, примитивной рекурсии и минимизации

Для более глубокого изучения класса функций, вычислимых на мащинах Тьюринга, нам необходимо ввести три операции над (вообще говоря, частичными) функциями.

С операцией *суперпозиции* мы уже познакомились в предыдущих главах. Здесь нам будет достаточно рассматривать суперпозицию довольно специального вида — «регулярную» суперпозицию. Пусть имеются функции

$$g_0(y_1, \ldots, y_m),\ g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n). \tag{4.2}$$

Образуем из них функцию f:

$$f(x_1, \ldots, x_n) = g_0(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)). \tag{4.3}$$

О функции f будем говорить, что она получена из функций (4.2) с помощью *операции суперпозиции* (или короче: *суперпозицией функций* (4.3)).

Поскольку в дальнейшем нам придется применять операцию суперпозиции к частично определенным функциям, необходимо уточнить, как действует операция суперпозиции в этом случае. Пусть функция $f(x_1, \ldots,$

x_n) получается из (частичных) функций (4.2) с помощью операции суперпозиции (4.3). Для любого набора $(a_1, \ldots, a_n)$ из N^n считаем значение $f(a_1, \ldots, a_n)$ определенным только в том случае, когда определены все значения

$$g_1(a_1, \ldots, a_n), \ldots, g_m(a_1, \ldots, a_n) \tag{4.4}$$

и определено значение функции g_0 на наборе с компонентами (4.4).

Перейдем к более сложной операции *примитивной рекурсии.* Пусть имеются (частично определенные) функции $g(x_1, \ldots, x_{n-1})$ и $h(x_1, \ldots, x_{n+1})$. Будем говорить, что функция $f(x_1, \ldots, x_n)$ получена из функций g, h с помощью операции *примитивной рекурсии* (по переменной x_n), если при всех значениях переменных $x_1, \ldots, x_n$ выполняются соотношения

$$\begin{cases} f(x_1, \ldots, x_{n-1}, 0) = g(x_1, \ldots, x_{n-1}), \\ f(x_1, \ldots, x_{n-1}, x_n + 1) = g(x_1, \ldots, x_{n-1}, x_n, f(x_1, \ldots, x_n)). \end{cases} \tag{4.5}$$

Так же, как в случае операции суперпозиции, необходимо пояснить, как будет действовать операция примитивной рекурсии на частичные функции. Для любого набора $(a_1, \ldots, a_n)$ из N^n значение $f(a_1, \ldots, a_n)$ считаем определенным только в том случае, когда определено значение $g(a_1, \ldots, a_{n-1})$ и для любого $b < a_n$ определено также значение

$$h(a_1, \ldots, a_{n-1}, b, f(a_1, \ldots, a_{n-1}, b))$$

(разумеется, значения $f(a_1, \ldots, a_{n-1}, b)$ при этом вычисляются согласно равенствам (4.5)).

Рассмотрим наиболее сильную операцию *минимизации.* Пусть $g(x_1, \ldots, x_n)$ — частично определенная функция. Определим функцию $f(x_1, \ldots, x_n)$ — результат применения операции минимизации μ к функции g,

$$f(x_1, \ldots, x_n) = (\mu y)(g(x_1, \ldots, x_{n-1}, y) = x_n). \tag{4.6}$$

Считаем, что для произвольного набора $(a_1, \ldots, a_n)$ из N^n значение $f(a_1, \ldots, a_n)$ определено и равно числу b в том и только том случае, когда:

1) значение $g(a_1, \ldots, a_{n-1}, b)$ определено и равно a_n;

2) при любом $z < b$ значение $g(a_1, \ldots, a_{n-1}, z)$ определено и отлично от a_n.

Понятно, что с помощью операции минимизации находится наименьшее (по y) решение уравнения

$$g(x_1, \ldots, x_{n-1}, y) = x_n.$$

Существенное ограничение здесь состоит в том, что в процессе поиска такого решения (если оно, конечно, имеется) мы не можем «перескакивать» через те точки y, в которых функция g не определена. Может показаться, что это ограничение носит искусственный характер. Однако без этого ограничения операция минимизации перестает быть эффективной операцией, а ее применение к алгоритмически вычислимой функции может привести к алгоритмически невычислимой функции.

Напомним, что в § 1 мы обосновали вычислимость функции I_i^n $(1 \leqslant i \leqslant n)$, равной значению своей i-й переменной. Совокупность всех функций I_i^n $(n = 1, 2, \ldots)$ обозначим через I.

В дальнейшем важную роль будет играть набор простейших вычислимых функций

$$0, \quad x + 1, \quad I, \tag{4.7}$$

где функцию-константу 0 можно считать зависящей от одной переменной. Введем одно из центральных понятий этой главы.

Назовем функцию *частично рекурсивной*, если ее можно получить из функций (4.7) с помощью (конечного числа применений) операций суперпозиции, примитивной рекурсии и минимизации. Класс всех частично рекурсивных функций обозначим через $\mathrm{F}_{\text{чр}}$.

Сделаем несколько замечаний по поводу введенного понятия. Прежде всего, как видно из самого названия, частично рекурсивная функция может быть частичной функцией (случай всюду определенной функции, разумеется, также не исключается). Далее, может сложиться впечатление, что имеются всюду определенные *рекурсивные* функции (такие функции действительно есть, они называются *общерекурсивными*), из которых частично рекурсивные функции получаются механическим «удалением» некоторых значений. Однако это не так. Частично рекурсивная функция — это единый объект, такой, например, как аналитическая функция. И получить частично рекурсивную функцию из «более простых» всюду определенных функций «удалением» значений, вообще говоря, невозможно.

Значение класса $\mathrm{F}_{\text{чр}}$ в математике вообще и в теории алгоритмов в частности определяется *тезисом Чёрча*.

Класс алгоритмически вычислимых (от натуральных аргументов) функций совпадает с классом $\mathrm{F}_{\text{чр}}$.

Детальное исследование класса $\mathrm{F}_{\text{чр}}$ мы отложим до последующих параграфов. А пока установим одно из важных соотношений в теории вы-

числимых функций.

Теорема 4.1. *Любая частично рекурсивная функция вычислима на машине Тьюринга.*

Доказательство проведем индукцией по построению класса частично рекурсивных функций.

Вычислимость исходных частично рекурсивных функций (4.7) обсуждалась в § 1. Остается показать, что операции суперпозиции, примитивной рекурсии и минимизации сохраняют вычислимость функций на машинах Тьюринга.

Итак, пусть функции $g_0, g_1, \ldots, g_m$ правильно вычислимы на машинах Тьюринга $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_m$, а функция $f(x_1, \ldots, x_n)$ получается из функций $g_0, g_1, \ldots, g_m$ с помощью операции суперпозиции (4.3). Будем предполагать, что все машины $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_m$ работают в алфавите $\{0, 1\}$. С принципиальной точки зрения вычислимость функции f не вызывает сомнений: достаточно последовательно с помощью машин $\mathcal{M}_1, \ldots, \mathcal{M}_m$ вычислить значения функций $g_1, \ldots, g_m$ (если, конечно, все вычисления заканчиваются) и затем к полученному набору значений применить машину Тьюринга $\mathcal{M}_0$. Однако при реализации этого плана возникают технические трудности, которые мы попытаемся обрисовать и преодолеть.

Первая трудность возникает сразу же, как только мы приступаем к вычислению значения функции g_1. Действительно, при вычислении значения $g_1(x_1, \ldots, x_n)$ основной код набора $(x_1, \ldots, x_n)$ будет, вообще говоря, утерян. Как после этого вычислять значения функций $g_2, \ldots, g_m$?

Для преодоления этой трудности нам придется воспользоваться дополнительными символами на ленте. Начнем с того, что несколько изменим процесс вычисления функций $g_1, \ldots, g_m$ на машинах $\mathcal{M}_1, \ldots, \mathcal{M}_m$. Именно, добавим к символам 0,1 ленточного алфавита машин $\mathcal{M}_1, \ldots, \mathcal{M}_m$ новый символ 2. Он будет играть роль дополнительного пустого символа, отмечая в процессе вычисления те клетки ленты, в которых хотя бы раз побывала головка машины Тьюринга и которые содержат символ 0.

В программе каждой из машин Тьюринга $\mathcal{M}_1, \ldots, \mathcal{M}_m$ заменим любую команду вида

$$aq_i \to 0Dq_j$$

командой

$$aq_i \to 2Dq_j.$$

После этого к каждой команде вида

$$0q_i \to bDq_j$$

добавим «дублирующую» команду

$$2q_i \to bDq_j.$$

Полученные в результате этих преобразований машины Тьюринга обозначим через $\mathcal{M}'_1, \ldots, \mathcal{M}'_m$.

Понятно, что машины $\mathcal{M}'_1, \ldots, \mathcal{M}'_m$ будут воспринимать символы 0 и 2 одинаково, как раньше воспринимали символ 0 машины $\mathcal{M}_1, \ldots, \mathcal{M}_m$. Главное отличие вычислений на машинах $\mathcal{M}'_1, \ldots, \mathcal{M}'_m$ от аналогичных вычислений на машинах $\mathcal{M}_1, \ldots, \mathcal{M}_m$ состоит в том, что после окончания вычисления (если оно действительно заканчивается) символом 2 будут отмечены все клетки ленты, где хотя бы раз побывала головка машины Тьюринга и которые в момент окончания вычисления являются пустыми (т.е. не содержат символ 1). Это свойство вычислений на машинах $\mathcal{M}'_1, \ldots, \mathcal{M}'_m$ поможет нам в дальнейшем создать основной код набора значений функций $g_1, \ldots, g_m$ для последующего применения машины $\mathcal{M}_0$.

Преобразование машин $\mathcal{M}_1, \ldots, \mathcal{M}_m$ в машины $\mathcal{M}'_1, \ldots, \mathcal{M}'_m$ пока не решает основную задачу: как на одной ленте организовать вычисление нескольких функций, сохраняя при этом результаты предыдущих вычислений. Для решения этой задачи служат так называемые *дорожки* на ленте машины Тьюринга.

Представим себе, что каждая клетка ленты машины Тьюринга разделена на m «этажей» так, что на каждом «этаже» можно записать один из символов 0,1,2. Части клеток ленты, принадлежащие одному и тому же «этажу», образуют *дорожку*. Введение дорожек, разумеется, представляет собой всего лишь наглядный технический прием, с помощью которого удобно описывать моделирование нескольких машин Тьюринга на одной ленте. Формально при введении дорожек мы переходим от алфавита $\{0, 1, 2\}$, в котором работают машины $\mathcal{M}'_1, \ldots, \mathcal{M}'_m$, к алфавиту B, состоящему из 3^m букв. Эти буквы можно представлять себе в виде наборов $(b_1, \ldots, b_m)$ длины m, где каждая компонента b_i есть буква алфавита $\{0, 1, 2\}$. Компоненты b_i букв $(b_1, \ldots, b_i, \ldots, b_m)$ как раз записываются на i-й дорожке ленты.

Имея теперь m дорожек на ленте (или, что эквивалентно, алфавит B из 3^m букв), нетрудно организовать процесс последовательного вычисления значений функций $g_1, \ldots, g_m$. Сначала основной код набора

$(x_1, \ldots, x_n)$ переводим на все m дорожек. При этом можно считать, что символам 0 и 1 исходного алфавита отвечают буквы $(0, \ldots, 0)$ и $(1, \ldots, 1)$ алфавита B. Затем на первой дорожке «запускаем» машину $\mathcal{M}'_1$. Она воспринимает только символы, записанные на первой дорожке, полностью игнорируя все, что записано на остальных $m-1$ дорожках. Если машина $\mathcal{M}'_1$ заканчивает вычисление, запускаем машину $\mathcal{M}'_2$ на второй дорожке. Вот здесь нам понадобятся символы 2, которые расставляет машина $\mathcal{M}'_1$ на первой дорожке. С их помощью (а также с помощью символов 1) мы отыскиваем начало основного кода на второй дорожке — достаточно посетить лишь те клетки, у которых первая дорожка содержит символы 1 или 2. Если машина $\mathcal{M}'_2$ заканчивает работу, то запускаем машину $\mathcal{M}'_3$ на третьей дорожке, и т.д.

Если значения всех функций $g_1, \ldots, g_m$ определены, то в результате описанного процесса моделирования машин $\mathcal{M}'_1, \ldots, \mathcal{M}'_m$ на m дорожках ленты будут записаны в основном коде значения

$$g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n). \tag{4.8}$$

Теперь необходимо «выровнять» эти записи, т.е. расположить их на соответствующих дорожках так, чтобы последовательность значений функций $g_1, \ldots, g_m$ выглядела как основной код набора (4.8). Здесь нам вновь понадобятся символы 2, которые могут находиться на любой из дорожек. Они упростят процесс «выравнивания» записей на дорожках, позволяя определить потенциальные границы этих записей.

Остается совсем немного для завершения вычисления значений функций $g_1, \ldots, g_m$. Следует только выполнить обратный переход из алфавита B в алфавит $\{0, 1\}$, т.е. сформировать на ленте основной код набора (4.8). Завершающий этап вычисления значения функции f состоит в том, что к полученному основному коду набора (4.8) применяется машина Тьюринга $\mathcal{M}_0$.

Подробно описанный алгоритм вычисления функции f на машине Тьюринга позволяет построить эту машину Тьюринга в виде композиции более простых машин Тьюринга, включающих, в частности, машины $\mathcal{M}_0, \mathcal{M}'_1, \ldots, \mathcal{M}'_m$.

Перейдем к операции примитивной рекурсии. Пусть функции g и h правильно вычислимы на машинах Тьюринга $\mathcal{M}_0, \mathcal{M}_1$, а функция $f(x_1, \ldots, x_n)$ получается из функций g, h с помощью операции примитивной рекурсии (4.5). Как и в случае операции суперпозиции, введем в ленточный алфавит машин $\mathcal{M}_0, \mathcal{M}_1$ символ 2, чтобы отмечать пустые клет-

ки, в которых в процессе вычисления побывали головки машин $\mathcal{M}_0, \mathcal{M}_1$. Соответствующим образом модифицированные машины Тьюринга обозначим через $\mathcal{M}'_0$ и $\mathcal{M}'_1$.

Для вычисления функции f нам понадобятся три дорожки на ленте. На первой дорожке будет постоянно находиться исходный набор значений переменных $x_1, \ldots, x_n$, на второй дорожке — «текущее» значение последней переменной (по которой ведется рекурсия), а на третьей дорожке будут вычисляться значения функций g и h. В связи с этим распределением дорожек первое действие машины Тьюринга, вычисляющей функцию f, будет состоять в переписывании основного кода набора $(x_1, \ldots, x_n)$ на первую дорожку и создании основного кода числа 0 на второй дорожке. Третью дорожку пока оставляем пустой.

Второй этап вычислений заключается в переносе с первой дорожки на третью чисел $x_1, \ldots, x_{n-1}$, формировании на третьей дорожке основного кода набора $(x_1, \ldots, x_{n-1})$ и применении к этому коду машины $\mathcal{M}'_0$. Все дальнейшие действия по вычислению значения функции f будут содержаться в цикле.

Итак, сравниваем число, записанное на второй дорожке, с числом x_n, имеющимся на первой дорожке. Если эти числа равны, то на третьей дорожке представлено в основном коде искомое значение функции f и нам необходимо лишь убрать дорожки и получить основной код результата вычислений. В противном случае пусть на второй и третьей дорожках записаны в основном коде числа y и z. Добавим 1 к числу y на второй дорожке, образуем на третьей дорожке основной код набора $(x_1, \ldots, x_{n-1}, y, z)$ и применим к этому набору машину $\mathcal{M}'_1$. После этого вернемся к началу цикла.

Операция минимизации рассматривается в значительной степени аналогично операции примитивной рекурсии. Поэтому соответствующие рассуждения мы опускаем. Теорема доказана.

УПРАЖНЕНИЯ

8. Описать подробно работу машины Тьюринга, которая осуществляет применение операции минимизации (4.6) к функции g.

§ 5. Универсальная машина Тьюринга

Универсальность — явление, характерное для теории алгоритмов. Большинство самых известных результатов теории алгоритмов так или иначе связаны с универсальностью.

Понятие универсальности проще всего объяснить на примере функций одной переменной. Пусть $f_0(x), f_1(x), \ldots$ — последовательность функций, заданных на N. Это могут быть, например, частично рекурсивные функции, но могут быть и всюду определенные невычислимые функции.

Функция $U(n,x)$ называется *универсальной для последовательности функций* $f_0, f_1, \ldots$, если выполняются два условия:

во-первых, для любого n из N функция $U(n,x)$ как функция от x совпадает с некоторой функцией $f_m(x)$;

во-вторых, для любой функции $f_m(x)$ найдется такое n (их может быть и несколько), что функция $U(n,x)$ как функция от x совпадает с функцией $f_m(x)$.

Это определение универсальной функции можно выразить несколько иначе, если сказать, что совокупность функций $\{U(0,x), U(1,x), \ldots\}$ совпадает с совокупностью $\{f_0(x), f_1(x), \ldots\}$ (подчеркнем еще раз, что в последовательности $U(0,x), U(1,x), \ldots$ функции $f_0(x), f_1(x), \ldots$ располагаются в произвольном порядке и, возможно, с повторениями).

Понятие универсальной машины Тьюринга можно считать вариантом понятия универсальной функции. Для простоты ограничимся только машинами Тьюринга, имеющими ленточный алфавит $\{0,1\}$.

Назовем машину Тьюринга $\mathcal{U}$ *универсальной*, если она вычисляет функцию $U(n,x)$, которая является универсальной для последовательности функций одной переменной, вычислимых на машинах Тьюринга с ленточным алфавитом $\{0,1\}$.

Теорема 4.2. *Универсальная машина Тьюринга существует.*

Доказательство. Мы продемонстрируем, как можно построить универсальную машину Тьюринга. К сожалению, нам не удастся выписать программу универсальной машины Тьюринга (в предлагаемом способе построения она будет содержать более сотни команд), однако принципы ее устройства довольно просты и мы постараемся их изложить.

Прежде всего, мы хотим занумеровать все машины Тьюринга, работающие в алфавите $\{0,1\}$, т.е. сопоставить каждой машине Тьюринга некоторое число из N. При этом способ нумерации должен быть таким, чтобы по номеру машины Тьюринга можно было бы (тоже на машине Тьюринга) однозначно восстановить ее программу.

Сначала закодируем команды машин Тьюринга. Команде (4.1) сопоставим слово в алфавите $\{0,1,2\}$, имеющее вид

$$2a_i2d(j)2a_l2d(D)2d(s)2, \tag{4.9}$$

где $d(m)$ — двоичное представление числа m и $d(L) = 0$, $d(R) = 1$, $d(S) = 01$.

Если программа машины Тьюринга $\mathcal{M}$ имеет p команд и им уже сопоставлены слова $w_1, \ldots, w_p$ вида (4.9) в алфавите $\{0, 1, 2\}$, то всей программе машиныв $\mathcal{M}$ сопоставим слово в алфавите $\{0, 1, 2, 3\}$, которое имеет вид

$$w_1 3 w_2 3 \ldots 3 w_p \tag{4.10}$$

(порядок слов $w_1, \ldots, w_p$ в слове (4.10) роли не играет). Номером машины $\mathcal{M}$ теперь будем считать число, имеющее представление (4.10) в четверичной системе счисления.

Понятно, что в результате каждая машина Тьюринга получит некоторый номер (и даже несколько номеров, поскольку слова $w_1, \ldots, w_p$ в слове (4.10) можно переставлять). При этом по номеру машины Тьюринга сравнительно просто восстановить программу машины: достаточно лишь представить этот номер в четверичной системе счисления. Отметим еще, что далеко не каждое натуральное число будет являться номером некоторой машины Тьюринга. Однако, пользуясь алгоритмом кодирования машин Тьюринга, не составляет особого труда проанализировать четверичное представление натурального числа и выяснить, определяет ли оно программу машины Тьюринга.

Перейдем теперь к построению универсальной машины Тьюринга $\mathcal{U}$. Собственно, мы проведем лишь описание функционирования универсальной машины $\mathcal{U}$. Детали построения программ для отдельных частей машины $\mathcal{U}$ мы оставляем читателю.

Итак, пусть на ленте машины $\mathcal{U}$ в основном коде представлена пара чисел n, x. Как и в § 4, выделим на ленте машины $\mathcal{U}$ три дорожки. Первая дорожка будет служить для записи программы машины Тьюринга, имеющей номер n (обозначим эту машину через $\mathcal{M}_n$). На второй дорожке будет храниться двоичный номер «текущего» состояния машины $\mathcal{M}_n$. А на третьей дорожке будет собственно моделироваться процесс применения машины $\mathcal{M}_n$ к аргументу x.

В соответствии с распределением роли дорожек машина $\mathcal{U}$ прежде всего переносит число n на первую дорожку и создает на ней четверичную запись n (при этом можно пользоваться «школьным» алгоритмом деления на 4 с остатком). На вторую дорожку машина $\mathcal{U}$ записывает число 1, а на третью — число x в основном коде. Кроме того, на третьей дорожке особым символом (меткой) помечается та клетка, которую в данный момент времени обозревает головка моделируемой машины $\mathcal{M}_n$.

Используя структуру представлений (4.9),(4.10), машина $\mathcal{U}$ на первой дорожке проверяет, является ли четверичная запись числа n номером некоторой машины Тьюринга. Если нет, то машина $\mathcal{U}$ реализует какую-либо простую функцию (например, константу 0).

Предположим, что на первой дорожке действительно записан код машины Тьюринга $\mathcal{M}_n$. В этом случае машина $\mathcal{U}$ шаг за шагом моделирует процесс применения машины $\mathcal{M}_n$ к аргументу x. Для этого с третьей дорожки считывается символ a_i, который в данный (моделируемый) момент времени обозревает головка машины $\mathcal{M}_n$. Затем машина $\mathcal{U}$ последовательно просматривает код машины $\mathcal{M}_n$, представленный на первой дорожке, и разыскивает в нем слово (4.9), где j — номер «текущего» состояния машины $\mathcal{M}_n$, записанный на второй дорожке. Найдя это слово, машина $\mathcal{U}$ осуществляет преобразования на второй и третьей дорожках ленты, соответствующие тройке (a_l, D, s). Именно, на второй дорожке двоичная запись числа j заменяется двоичной записью числа s, а на третьей дорожке обозреваемый машиной $\mathcal{M}_n$ символ a_i заменяется символом a_l и сдвигается метка, указывающая положение головки машины $\mathcal{M}_n$ на ленте. Далее машина $\mathcal{U}$ переходит к моделированию следующего шага работы машины $\mathcal{M}_n$.

Если машина $\mathcal{M}_n$ заканчивает работу над аргументом x, то машина $\mathcal{U}$ преобразует результат вычислений машины $\mathcal{M}_n$, полученный на третьей дорожке, в основной код. В противном случае машина $\mathcal{U}$, как и машина $\mathcal{M}_n$, работает неограниченно долго. Теорема доказана.

Как мы отмечали выше, существование универсальной машины Тьюринга $\mathcal{U}$ равносильно существованию вычислимой функции $U(n, x)$, универсальной для множества всех вычислимых функций одной переменной. В свою очередь из последнего факта вытекает ряд следствий, составляющих ядро теории алгоритмов. Приведем некоторые из них.

Рассмотрим прежде всего «диагональную» функцию $U(x, x)+1$. Очевидно, что она вычислима. Значит, для некоторого n и всех x должно выполняться равенство

$$U(n, x) = U(x, x) + 1.$$

Подставляя в него вместо x число n, получим «противоречивое» равенство

$$U(n, n) = U(n, n) + 1. \tag{4.11}$$

Однако противоречия здесь нет. Равенство (4.11) показывает лишь, что

значение $U(n,n)$ не определено (из построения функции U легко понять, что функция U частичная).

Таким образом, имея универсальную вычислимую функцию U, мы может точно указать набор (именно набор (n,n), где n есть номер функции $U(x,x)+1$ в нумерации частично рекурсивных функций, даваемой функцией U), на котором функция U заведомо не определена. Итак, функция $U(x,x)+1$ вычислима и не всюду определена.

Однако бывают такие вычислимые не всюду определенные функции (например функция $[\log_2 x]$), которые легко доопределяются до всюду определенных вычислимых функций. Функция $U(x,x)+1$ этим свойством не обладает.

Следствие 1. *Не существует такой всюду определенной вычислимой функции, которая совпадает с функцией $U(x,x)+1$ на всей ее области определения.*

Доказательство. Предположим, напротив, что такая функция $V(x)$ существует. Тогда для некоторого n и всех x выполняется равенство

$$V(x) = U(n,x). \tag{4.12}$$

В частности, оно верно при $x=n$. Однако функция V по предположению всюду определена. Следовательно, для данного n значение $U(n,n)$ определено и совпадает со значением

$$V(n) = U(n,n)+1.$$

Противоречие показывает, что наше предположение неверно, и следствие доказано.

Может сложиться впечатление, что функцию $U(x,x)+1$ нельзя доопределить до всюду определенной вычислимой функции вследствие того, что она «непредсказуемым образом» принимает «очень большие» и «очень малые» значения. В следствии 2 мы рассматриваем функцию, которая принимает лишь значения 0 и 1.

Следствие 2. *Не существует такой всюду определенной вычислимой функции, которая является доопределением функции $\overline{\mathrm{sg}}(U(x,x))$.*

Доказательство. Предположим, что такая функция $V(x)$ существует. Вновь выбираем такое число n, что для всех x выполняется равенство

(4.12). Поскольку V — всюду определенная функция, при $x = n$ получаем из него невозможное равенство

$$U(n, n) = \overline{\mathrm{sg}}(U(n, n)).$$

Следствие доказано.

И еще один интересный результат связан с функцией U. Рассмотрим множество M всех тех чисел n, для которых функция $U(n, x)$ как функция от x всюду определена. Существует ли «пересчет» (возможно, с повторениями) множества M с помощью какой-либо всюду определенной вычислимой функции? Оказывается, нет.

В самом деле, предположим, что такая всюду определенная функция $f(x)$ существует. Рассмотрим вычислимую функцию

$$V(x) = U(f(x), x) + 1. \tag{4.13}$$

Поскольку функция f принимает лишь значения из множества M, функция V будет всюду определенной вычислимой функцией. Покажем, что она отличается от любой всюду определенной вычислимой функции $g(x)$. Действительно, согласно определению универсальной функции найдется такое n, что для всех x выполняется равенство

$$g(x) = U(n, x).$$

Пусть число m таково, что $f(m) = n$. Тогда из соотношения (4.13) следует, что функция V отличается от функции g в точке $x = m$. Получили противоречие. Значит, множество M невозможно перечислить всюду определенной вычислимой функцией.

УПРАЖНЕНИЯ

9. Доказать, что всякая вычислимая функция имеет в нумерации U бесконечное число номеров.

10. Доказать, что функция $U(x, 2x)$ не имеет всюду определенного вычислимого доопределения.

11. Пусть M есть множество всех тех n, для которых функция $U(n, x)$ как функция от x определена хотя бы в одной точке. Доказать, что множество M можно перечислить подходящей всюду определенной вычислимой функцией.

§ 6. Классы P и NP

Пусть A, B — конечные алфавиты, f — функция вида $A^* \to B^*$, $T(n)$ — функция вида $N \to N$. Будем говорить, что функция f *вычислима за время* $T(n)$, если существует машина Тьюринга $\mathcal{M}$ (алфавит машины $\mathcal{M}$ включает алфавиты A и B), которая вычисляет функцию f и при этом для любого слова $\bar{a} \in A^*$ длины n время вычисления значения $f(\bar{a})$ на машине $\mathcal{M}$ (т.е. число шагов, которое машина $\mathcal{M}$ затрачивает на получение результата $f(\bar{a})$) не превосходит $T(n)$.

Говорим, что функция f *вычислима за полиномиальное время* (или *полиномиально вычислима*), если существуют машина Тьюринга $\mathcal{M}$ и полином $p(n)$ с натуральными коэффициентами, такие, что машина $\mathcal{M}$ вычисляет функцию f за время $p(n)$.

В этом и следующем параграфах мы будем рассматривать *задачи распознавания* множеств (языков) $L \subseteq A^*$, под которыми будем понимать задачи вычисления *характеристических функций* $f_L : A^* \to \{0, 1\}$, где при любом $\bar{a}$ из A^*

$$\bar{a} \in L \Leftrightarrow f_L(\bar{a}) = 1.$$

Обозначим через P класс всех множеств, характеристические функции которых вычислимы за полиномиальное время (полиномиально распознаваемые множества).

Пусть $L_1 \subseteq A^*$ и $L_2 \subseteq B^*$. Говорим, что множество L_1 *полиномиально сводится* (или P-*сводится*) к множеству L_2, если существует полиномиально вычислимая функция $f : A^* \to B^*$, такая, что

$$(\forall \bar{a})(\bar{a} \in L_1 \Leftrightarrow f(\bar{a}) \in L_2).$$

Отметим, что отношение полиномиальной сводимости рефлексивно и транзитивно.

Утверждение 4.1. *Если множество* L_1 *полиномиально сводится к множеству* L_2 *и* $L_2 \in$ P, *то* $L_1 \in$ P.

Доказательство. Пусть функция f_1 осуществляет полиномиальное сведение множества L_1 к множеству L_2, а f_2 — полиномиально вычислимая характеристическая функция множества L_2. Из определений следует, что $f_2(f_1(x))$ есть характеристическая функция множества L_1. Остается проверить, что функция $f_2(f_1(x))$ полиномиально вычислима. Однако если машина Тьюринга $\mathcal{M}_1$ вычисляет функцию f_1 за время $p_1(n)$,

а машина Тьюринга $\mathcal{M}_2$ — функцию f_2 за время $p_2(n)$, где p_1, p_2 — полиномы, то композиция $\mathcal{M}_2(\mathcal{M}_1)$ будет вычислять функцию $f_2(f_1)$ за время $p_1(n) + p_2(n + p_1(n))$. Утверждение доказано.

Для определения класса NP нам понадобится ввести понятие *недетерминированной машины Тьюринга*. С недетерминированными конечными автоматами мы уже встречались в главе 2. В недетерминированных машинах Тьюринга (сокращенно НМТ) происходит дальнейшее развитие идеи «недетерминизма», заложенной в понятии недетерминированного конечного автомата.

Так же, как недетерминированный конечный автомат, НМТ на каждом шаге работы может выбрать несколько (но конечное число) вариантов продолжения вычисления. Однако в отличие от конечного автомата НМТ способна не только переходить в различные состояния, но также совершать различные замены обозреваемого символа и выполнять различные движения головкой на ленте. Иными словами, в каждый момент времени НМТ может по-разному преобразовывать как свою внутреннюю память (внутренние состояния), так и внешнюю память (содержимое ленты и положение головки на ней). Разумеется, эта возможность чисто виртуальная, и реальные вычислительные устройства такими возможностями не обладают. Стоит еще отметить, что вычислять функции на НМТ, вообще говоря, нельзя. Так же, как для недетерминированных конечных автоматов, здесь можно говорить лишь о множествах слов, допускаемых (распознаваемых) НМТ.

Перейдем к более формальному определению недетерминированной машины Тьюринга. Как и обычная машина Тьюринга, НМТ $\mathcal{M}$ имеет бесконечную ленту, разбитую на клетки, головку, способную перемещаться по ленте и читать написанные на ней символы, и управляющее устройство. Мы сохраняем обозначения $A = \{a_0, a_1, \ldots, a_k\}$ и $Q = \{q_0, q_1, \ldots, q_r\}$ за ленточным алфавитом машины и множеством ее (внутренних) состояний. В отличие от (детерминированной) машины Тьюринга машина $\mathcal{M}$ может иметь в своей программе несколько различных команд (4.1) с одинаковыми левыми частями $a_i q_j$. Как же «вычисляет» машина $\mathcal{M}$?

В начальный момент времени на ленту машины записывается слово $\bar{a}$ в алфавите $\{a_1, \ldots, a_k\}$, машина устанавливается в состояние q_1, а ее головка — на первый символ слова $\bar{a}$. Далее «выполняется» одна из подходящих команд (4.1). Это значит, что в следующий момент времени потенциально может возникнуть одна из *конфигураций* (слово на ленте,

положение головки на слове и состояние машины), определяемых правыми частями команд (4.1). Мы не можем сказать точно, какая именно конфигурация будет реализована в данный момент времени, но вынуждены рассматривать их все «одновременно». Если мы выберем одну из таких конфигураций K, то на следующем шаге придем точно в такое же положение: к конфигурации K можно будет применить, вообще говоря, несколько команд (4.1), что влечет за собой появление нескольких новых конфигураций, и т.д.

Иногда процесс применения машины $\mathcal{M}$ к слову иллюстрируют таким образом. Считают, что в тот момент, когда машине $\mathcal{M}$ необходимо применить одну из нескольких возможных команд, для каждой из таких команд возникает своя копия машины $\mathcal{M}$, которая продолжает работать далее подобно исходной машине $\mathcal{M}$. Понятно, что такое «размножение» машин Тьюринга приводит, вообще говоря, к неограниченному росту числа копий машины $\mathcal{M}$. Однако некоторое преимущество этой модели состоит в том, что сохраняется «присутствие» машины $\mathcal{M}$ в любой момент времени и в каждой возможной конфигурации.

Из приведенного выше описания видно, что вычисление на недетерминированной машине Тьюринга выглядит не как линейно упорядоченная цепочка конфигураций (что характерно для детерминированных вычислительных устройств), а скорее как дерево, в котором могут присутствоать конечные и бесконечные ветви. Конечные ветви (их может быть несколько) соответствуют достижению машиной $\mathcal{M}$ заключительного состояния q_0, бесконечные — отсутствию состояния q_0 в конфигурациях ветви.

Так же, как для конечных автоматов, определяем для НМТ $\mathcal{M}$ множество $D(\mathcal{M})$ — совокупность всех слов $\bar{a}$ в алфавите $\{a_1, \ldots, a_k\}$, для которых в дереве конфигураций, построенном по слову $\bar{a}$, имеется хотя бы одна конечная ветвь. Иными словами, $D(\mathcal{M})$ состоит в точности из всех слов $\bar{a}$, для которых машина $\mathcal{M}$ может «выбрать» вычисление, завершающееся в заключительном состоянии q_0. Если $\bar{a} \in D(\mathcal{M})$, то говорим, что машина $\mathcal{M}$ *допускает* (*распознает*) слово $\bar{a}$. Множество $D(\mathcal{M})$ при этом называют множеством, *допускаемым* (*распознаваемым*) машиной $\mathcal{M}$.

Пусть $T(n)$ — функция типа $N \to N$ и $L = D(\mathcal{M})$. Будем говорить, что машина $\mathcal{M}$ допускает (распознает) множество L *за время* $T(n)$, если для любого слова $\bar{a} \in L$ длины n существует вычисление машины $\mathcal{M}$, допускающее слово $\bar{a}$, длина которого не превосходит $T(n)$. Иными

словами, в дереве конфигураций машины $\mathcal{M}$, отвечающем слову $\bar{a}$, существует конечная ветвь (с заключительным состоянием q_0) длины не более $T(n)$.

Обозначим через NP класс всех множеств, распознаваемых на недетерминированных машинах Тьюринга за полиномиальное время.

Заметим, что множества из класса NP можно распознавать и на детерминированных машинах Тьюринга. Однако в общем случае не известны (неперебoрные) алгоритмы распознавания, которые приводили бы к времени распознавания, существенно меньшему, чем экпоненциальное.

Приведем примеры нескольких известных множеств из класса NP. Нижеследующие множества из класса NP мы по традиции будем формулировать в виде *задач*: объекты, служащие решениями задачи, образуют искомое множество.

Напомним, что конъюнктивной нормальной формой (сокращенно КНФ) называется булева формула вида $D_1 \,\&\, D_2 \,\&\, \ldots \,\&\, D_l$, где каждая формула D_i имеет вид $t_{i1} \vee t_{i2} \vee \ldots \vee t_{in_i}$, t_{is} — пременная либо отрицание переменной, причем каждая переменная встречается в дизъюнкции D_i не более одного раза. Выражения t_{is} называются *литералами*.

Первой (и, видимо, самой известной) задачей будет задача ВЫПОЛНИМОСТЬ (сокращенно ВЫП).

Вход: произвольная формула $F(x_1, \ldots, x_m)$ в виде КНФ.

Вопрос: является ли формула F *выполнимой*, т.е. существует ли такой набор $(a_1, \ldots, a_m)$ из E_2^m, что $F(a_1, \ldots, a_m) = 1$?

Утверждение 4.2. *Задача* ВЫП *принадлежит классу* NP.

Доказательство. Поскольку формула F может содержать произвольное число переменных, необходимо условиться о способе кодирования формул словами конечного алфавита. В качестве кодирующего алфавита возьмем алфавит $A = \{0, 1, x, -, \vee, \&, (,)\}$. Переменную x_i будем задавать символом x с последующей двоичной записью числа i. На ленте (распознающей НМТ) могут быть записаны произвольные слова в алфавите A. Поэтому прежде чем приступить собственно к распознаванию выполнимости формулы, машине Тьюринга следует определить, является ли записанное слово кодом некоторой КНФ. Нетрудно видеть, что эту проверку можно выполнить детерминированным образом и за полиномиальное время.

Предположим, что на ленте машины Тьюринга помимо записи формулы F имеется некоторый набор $(a_1, \ldots, a_m)$ значений ее переменных.

Понятно, что в этом случае сравнительно просто (детерминированно и за полиномиальное время) можно проверить, выполняет ли набор $(a_1, \ldots, a_m)$ формулу F. Таким образом, задача сводится к тому, чтобы недетерминированно «угадать» набор, выполняющий формулу F. Заметим, что такого набора может и не быть.

Итак, НМТ, используя запись формулы F, сначала детерминированно «отмеряет» m клеток на ленте для записи предполагаемого набора $(a_1, \ldots, a_m)$. Затем, проходя по этому массиву из m (пустых) клеток, записывает там произвольные символы 0 и 1. Это — единственный этап в работе НМТ, когда по существу необходимо воспользоваться недетерминированностью машины. Здесь на каждом шаге движения по массиву из m клеток машина недетерминированно записывает в клетку один из символов 0 или 1. Закончив этот недетерминированный этап, машина переходит к проверке выполнимости формулы F на выписанном двоичном наборе. Утверждение доказано.

В качестве следующей задачи рассмотрим задачу КЛИКА.

Вход: произвольный неориентированный граф G и натуральное число k.

Вопрос: существует ли в графе G клика размера k (т.е. полный подграф с k вершинами).

В этой задаче в качестве кодирующего алфавита можно взять, например, алфавит $A = \{0, 1, ;\}$, граф G можно задать матрицей смежности, а затем записать матрицу одним словом, отделяя строки разделителем ;. Число k задается своим двоичным представлением.

Основная часть работы НМТ сводится к недетерминированному формированию списка из k вершин графа G и последующей проверки того, что данные вершины образуют полный подграф графа G.

Близкой к задаче КЛИКА является задача ГАМИЛЬТОНОВ ЦИКЛ.

Вход: произвольный неориентированный граф G.

Вопрос: существует ли в графе G гамильтонов цикл (т.е. цикл, проходящий через каждую вершину графа ровно один раз).

УПРАЖНЕНИЯ

12. Доказать, что следующие множества принадлежат классу P:

1) множество значений произвольного полинома $p(x)$ с натуральными коэффициентами (числа на ленте машины Тьюринга представляются в двоичной записи);

2) множество решений уравнения $p_1(x_1, \ldots, x_n) = p_2(x_1, \ldots, x_n)$ в области N^n, где p_1, p_2 — полиномы с натуральными коэффициентами.

3) множество значений функции c^x, где c — натуральное число, большее 1;

4) множество квадратных (0,1)-матриц с определителем, равным 1 (сложение элементов выполняется по модулю 2).

13. Доказать принадлежность классу P следующих множеств:

1) множество нелинейных булевых функций, заданных таблицами своих значений;

2) множество немонотонных булевых функций, заданных таблицами своих значений либо полиномами Жегалкина;

3) множество несамодвойственных булевых функций, заданных таблицами своих значений либо совершенными ДНФ.

14. Доказать, что классу NP принадлежат следующие задачи:

1) существование двоичного набора, на котором заданная ДНФ обращается в нуль;

2) нелинейность булевой функции, заданной в виде ДНФ;

3) немонотонность булевой функции, заданной в виде ДНФ;

4) несамодвойственность булевой функции, заданной в виде ДНФ.

15. Доказать принадлежность классу NP следующих задач:

1) ВЕРШИННОЕ ПОКРЫТИЕ.

Вход: граф $G = (V, E)$ и натуральное число l.

Вопрос: существует ли такое такое множество $R \subseteq V$, что $|R| \leqslant l$ и каждое ребро графа G инцидентно некоторой вершине из R.

2) ПОКРЫТИЕ МНОЖЕСТВ.

Вход: семейство $F = \{S_1, \ldots, S_m\}$ подмножеств множества S, где $S_1 \cup \ldots \cup S_m = S$, и натуральное число h.

Вопрос: существует ли такое такое подсемейство $T \subseteq S$, что $|T| \leqslant h$ и $\cup S_j = S$, где объединение берется по всем множествам S_j из T.

3) РАСКРАСКА.

Вход: граф $G = (V, E)$ и натуральное число k.

Вопрос: существует ли такая функция $\chi : V \to \{1, 2, \ldots, k\}$, что $\chi(u) \neq \chi(v)$ для любого ребра (u, v) из E.

§ 7. NP-полнота. Теорема Кука

Нетрудно видеть, что имеет место включение P $\subseteq$ NP. Является ли это включение собственным? Этот вопрос представляет собой одну из

наиболее интересных проблем современной теории сложности вычислений. Несмотря на многочисленные попытки решить сформулированную проблему, предпринятые в течение более чем 40 лет, сколько-нибудь существенных продвижений в этом направлении пока не получено. Большинство специалистов склоняются к мнению, что $\mathrm{P} \neq \mathrm{NP}$. В связи с этим вызывает интерес определение в классе NP некоторых «ключевых» задач — задач, которые «содержат» в себе все задачи из NP. Используя понятие P-сводимости, мы хотим для этих целей ввести понятие NP-*полноты*.

Назовем множество L NP-*трудным*, если любое множество из класса NP P-сводится к множеству L. Назовем множество L NP-*полным*, если L NP-*трудно* и $L \in \mathrm{NP}$.

Возникает вопрос: существуют ли вообще NP-полные множества? Ответ на этот вопрос дает следующая теорема С. Кука [5, 4].

Теорема 4.3. *Задача* ВЫП *является* NP-*полной.*

Доказательство. Включение ВЫП $\in$ NP установлено в утверждении 4.2. Поэтому остается показать, что любое множество L из NP полиномиально сводится к множеству ВЫП.

Пусть L есть множество слов в алфавите $A_1 = \{a_1, \ldots, a_k\}$ и $L \in \mathrm{NP}$. Тогда найдутся такая недетерминированная машина Тьюринга $\mathcal{M}$ и полином p с натуральными коэффициентами, что машина $\mathcal{M}$ распознает множество L за время $p(n)$. Иными словами, если $\bar{a} \in L$ и слово $\bar{a}$ имеет длину n, то для $\bar{a}$ существует допускающее вычисление машины $\mathcal{M}$ длины не более $p(n)$. Если же $\bar{a} \notin L$, то допускающего вычисления для слова $\bar{a}$ не существует.

Покажем, что множество L P-сводится к множеству ВЫП. Пусть B — алфавит задачи ВЫП. Нам необходимо определить такую полиномиально вычислимую функцию $\varphi : \ A_1^* \to B^*$ (значением которой всегда является КНФ), что

$$\bar{a} \in L \Leftrightarrow \varphi(\bar{a}) = F_{\bar{a}} - \text{выполнимая КНФ}.$$

Будем предполагать, что в начальный момент времени на ленте машины $\mathcal{M}$ записано слово $\bar{a}$ (остальная часть ленты заполнена символами a_0), машина находится в состоянии q_1 и головка машины установлена на первую букву слова $\bar{a}$. Пусть $n = |\bar{a}|$. Тогда в случае $\bar{a} \in L$ у машины $\mathcal{M}$ имеется вычисление, приводящее в заключительное состояние q_0 за не более чем $p(n)$ шагов. Если же $\bar{a} \notin L$, то никакое вычисление машины $\mathcal{M}$ не переводит ее в заключительное состояние.

Преобразуем машину $\mathcal{M}$ таким образом, чтобы после попадания в заключительное состояние она оставалась бы там постоянно, не меняя содержимого ленты и не сдвигая головку. Формально по достижении заключительного состояния машина $\mathcal{M}$ продолжает работу бесконечно долго. Тогда для любого слова $\bar{a} \in A_1^*$ длины n будем иметь

$\bar{a} \in L \Leftrightarrow$ машина $\mathcal{M}$, начиная работу на слове $\bar{a}$, в момент времени $p(n)$ будет находиться в состоянии q_0. (4.14)

Наша цель — записать правую часть эквивалентности (4.14) в виде КНФ $F_{\bar{a}}$ так, чтобы формула $F_{\bar{a}}$ была выполнимой в том и только том случае, когда истинна правая часть этой эквивалентности.

Напомним, что конфигурацией машины $\mathcal{M}$ в момент времени t мы назвали тройку, состоящую из слова, записанного на ленте в момент t, состояния машины $\mathcal{M}$ в момент t и положения ее головки в этот момент времени. Пусть K_t обозначает конфигурацию машины $\mathcal{M}$ в момент t. Запишем эквивалентность (4.14) в более подробной форме:

$\bar{a} \in L \Leftrightarrow (\exists K_0)(\exists K_1)\ldots(\exists K_{p(n)})(K_0$ — начальная конфигурация машины $\mathcal{M}$ для слова $\bar{a}$, конфигурация $K_{p(n)}$ содержит заключительное состояние q_0 и для всякого j $(0 \leqslant j \leqslant p(n)-1)$ конфигурация K_{j+1} получается из конфигурации K_j за один такт работы машины $\mathcal{M})$. (4.15)

Пусть клетки на ленте машины $\mathcal{M}$ занумерованы целыми числами слева направо, причем клетка, в которой машина $\mathcal{M}$ находится в начальный момент времени, имеет номер 0. Тогда за $p(n)$ шагов работы машины $\mathcal{M}$ головка может побывать только в клетках с номерами от $-p(n)$ до $p(n)$. Поэтому в конфигурациях $K_0, K_1, \ldots, K_{p(n)}$ будут учитываться только эти клетки.

Введем булевы переменные $x_{i,j}^t,\ y_i^t,\ z_l^t$, где

$$-p(n) \leqslant i \leqslant p(n), \quad 0 \leqslant j \leqslant k, \quad 0 \leqslant l \leqslant r, \quad 0 \leqslant t \leqslant p(n).$$

Содержательно этим переменным мы придадим следующий смысл:

$(x_{i,j}^t = 1) \Leftrightarrow$ i-я клетка конфигурации K_t содержит букву a_j;
$(y_i^t = 1) \Leftrightarrow$ в конфигурации K_t головка машины обозревает клетку с номером i;
$(z_l^t = 1) \Leftrightarrow$ конфигурация K_t содержит состояние q_l.

Искомая формула $F_{\bar{a}}$ будет зависеть от всех переменных $x_{i,j}^t,\ y_i^t,\ z_l^t$. Мы хотим, чтобы формула $F_{\bar{a}}$ принимала значение 1 на некотором двоичном наборе тогда и только тогда, когда

1) данный набор корректно задает последовательность конфигураций $K_0, K_1, \ldots, K_{p(n)}$ машины Тьюринга $\mathcal{M}$;

2) конфигурация K_0 является правильной начальной конфигурацией для слова $\bar{a}$;

3) конфигурация $K_{p(n)}$ содержит состояние q_0;

4) для всякого j $(0 \leqslant j \leqslant p(n)-1)$ конфигурация K_{j+1} получается из конфигурации K_j согласно программе машины $\mathcal{M}$ за один такт работы.

Рассмотрим условие 1. Нетрудно видеть, что его можно выразить следующим образом: при любом t $(0 \leqslant t \leqslant p(n))$

для любого i ровно одна из переменных $x_{i,j}^t$ принимает значение 1,
ровно одна из переменных y_i^t принимает значение 1,
ровно одна из переменных z_l^t принимает значение 1. (4.16)

Обозначим через $h(v_1, \ldots, v_s)$ булеву функцию, принимающую значение 1 только на наборах с одной единичной компонентой. Имеем

$$h(v_1, \ldots, v_s) = (v_1 \vee \ldots \vee v_s) \,\&\, \left(\mathop{\&}_{i \neq j} (\bar{v}_i \vee \bar{v}_j) \right).$$

Длина КНФ этой функции (число вхождений переменных) равна s^2. С помощью функции h выразим условие (4.16) в виде КНФ:

$$F_1 = \mathop{\&}_{0 \leqslant t \leqslant p(n)} ((\mathop{\&}_{-p(n) \leqslant i \leqslant p(n)} h(x_{i,0}^t, x_{i,1}^t, \ldots, x_{i,k}^t)) \&$$

$$\& h(y_{-p(n)}^t, y_{-p(n)+1}^t, \ldots, y_{p(n)}^t) \,\&\, h(z_0^t, z_1^t, \ldots, z_r^t)).$$

Учитывая сложность КНФ функции h, получаем, что длина КНФ формулы F_1 равна

$$(p(n)+1)((2p(n)+1)(k+1)^2 + (2p(n)+1)^2 + (r+1)^2),$$

т.е. является полиномом от n (k и r суть константы, зависящие только от машины $\mathcal{M}$).

Перейдем к условию 2. Будем предполагать, что набор переменных $x_{i,j}^t$, y_i^t, z_l^t корректно задает конфигурацию K_0. Пусть $\bar{a} = a_{j_1} a_{j_2} \ldots a_{j_n}$. Тогда условие 2 можно выразить следующей КНФ:

$$F_2 = x_{0,j_1}^0 \,\&\, x_{1,j_2}^0 \,\&\, \ldots \,\&\, x_{n-1,j_n}^0 \,\&\, \left(\mathop{\&}_{-p(n) \leqslant i \leqslant -1} x_{i,0}^0 \right) \,\&\, \left(\mathop{\&}_{n \leqslant i \leqslant p(n)} x_{i,0}^0 \right) \,\&\, y_0^0 \,\&\, z_1^0.$$

Формула F_2 имеет сложность $2p(n)+3$.

Условие 3 выражается КНФ из одной переменной: $z_0^{p(n)}$.

Рассмотрим условие 4. Команды машины $\mathcal{M}$ представим в виде

$$a_j q_l \to \{a_{\sigma(j,l)} D(j,l) q_{\tau(j,l)}\},$$

где посредством $a_{\sigma(j,l)} D(j,l) q_{\tau(j,l)}$ обозначена «общая» правая часть команды с левой частью $a_j q_l$ (напомним, что машина $\mathcal{M}$ недетерминированна) и $D(j,l) = -1, 0, 1$ в зависимости от движения головки машины $\mathcal{M}$ на ленте. Считая, что набор переменных $x_{i,j}^t,\ y_i^t,\ z_l^t$ корректно задает конфигурации $K_0, K_1, \ldots, K_{p(n)}$, запишем условие 4 в виде следующей формулы:

$$F_4' = \mathop{\&}_{0 \leqslant t \leqslant p(n)-1} \ \mathop{\&}_{-p(n) \leqslant i \leqslant p(n)} \ \mathop{\&}_{0 \leqslant j \leqslant k} \ \mathop{\&}_{0 \leqslant l \leqslant r} ((x_{i,j}^t \,\&\, y_i^t \,\&\, z_l^t \Rightarrow$$

$$\vee x_{i,\sigma(j,l)}^{t+1} \,\&\, y_{i+D(j,l)}^{t+1} \,\&\, z_{\tau(j,l)}^{t+1}) \,\&\, (\bar{y}_i^t \Rightarrow \mathop{\&}_{0 \leqslant j \leqslant k} (x_{i,j}^{t+1} \sim x_{i,j}^t))),$$

где дизъюнкция во второй строке формулы берется по всем командам, имеющим левую часть $a_j q_l$. Первая импликация этой формулы показывает, какими в момент $t+1$ будут символы на ленте, движение головки и новое состояние, вторая — что символы во всех остальных клетках остаются без изменения. Первая импликация зависит не более чем от $3((k+1)(r+1)+1)$ переменных, вторая — от $2k+3$ переменных. Каждую из них можно привести к КНФ, суммарная сложность C этих КНФ будет зависеть только от k, r. В результате формула F_4' преобразуется в КНФ F_4 длины

$$Cp(n)(2p(n)+1)(k+1)(r+1).$$

Положим

$$F_{\bar{a}} = F_1 \,\&\, F_2 \,\&\, F_3 \,\&\, F_4.$$

Тогда $F_{\bar{a}}$ — КНФ, которая на наборе переменных $(\{x_{i,j}^t\}, \{y_i^t\}, \{z_l^t\})$ принимает значение 1 тогда и только тогда, когда для входного слова $\bar{a}$ набор значений переменных корректно определяет вычисление на машине $\mathcal{M}$, приводящее к заключительному состоянию q_0. Длина формулы $F_{\bar{a}}$ ограничена сверху некоторым полином от n (напомним, что $n = |\bar{a}|$). При этом нетрудно заметить, что сама формула $F_{\bar{a}}$ эффективно определяется по программе машины $\mathcal{M}$ и выписывается за время, ограниченное полиномом от ее длины, т.е. полиномом от длины слова $\bar{a}$. Таким образом, отображение $\bar{a} \to F_{\bar{a}}$ определяет полиномиальное сведение множества L к множеству ВЫП. Значит, ВЫП — NP-трудная задача. А поскольку

ВЫП $\in$ NP, приходим к заключению, что ВЫП — NP-полная задача. Теорема доказана.

Конъюнктивную нормальную форму, у которой любая дизъюнкция содержит не более трех слагаемых, назовем 3-КНФ. Задача 3-ВЫПОЛНИМОСТЬ (сокращенно 3-ВЫП) представляет собой вариант задачи ВЫПОЛНИМОСТЬ, когда множество рассматриваемых КНФ ограничено 3-КНФ.

Теорема 4.4. *Задача* 3-ВЫП *является* NP-*полной.*

Доказательство. Ввиду утверждения 4.2 задача 3-ВЫП принадлежит классу NP. Мы покажем, что задача ВЫП P-сводится к задаче 3-ВЫП. Тогда в силу теоремы 4.3 и транзитивности P-сводимости любое множество из класса NP будет P-сводится к множеству 3-ВЫП, т.е. задача 3-ВЫП окажется NP-полной.

Покажем, как по произвольной КНФ K можно эффективно и полиномиально (по времени) определить 3-КНФ $\varphi(K)$ так, что будет справедлива эквивалентность

$$K \text{ выполнима } \Leftrightarrow \varphi(K) \text{ выполнима.}$$

Пусть $K = D_1 \& \ldots \& D_s$, где $D_1, \ldots, D_s$ — дизъюнкции от переменных $x_1, \ldots, x_n$. Если дизъюнкция D_i содержит не более трех переменных, то оставляем ее без изменения. Пусть теперь $D_i = t_1 \vee t_2 \vee \ldots \vee t_m$, где $t_1, \ldots, t_m$ — литералы и $m \geqslant 4$. Положим

$$F_i = (t_1 \vee t_2 \vee y_1) \,\&\, (\bar{y}_1 \vee t_3 \vee y_2) \,\&\, (\bar{y}_2 \vee t_4 \vee y_3) \,\&\, \ldots$$

$$\ldots \,\&\, (\bar{y}_{m-4} \vee t_{m-2} \vee y_{m-3}) \,\&\, (\bar{y}_{m-3} \vee t_{m-1} \vee t_m),$$

где $y_1, \ldots, y_{m-3}$ — переменные, отличные от переменных $x_1, \ldots, x_n$. Заметим, что сложность КНФ F_i не более чем в три раза превосходит сложность дизъюнкции D_i.

Покажем, что из равенства $F_i = 1$ следует равенство $D_i = 1$. Пусть $F_i(\tilde{a}, \tilde{b}) = 1$, где $\tilde{a} = (a_1, \ldots, a_n)$, $\tilde{b} = (b_1, \ldots, b_{m-3})$. Если $b_1 = 0$, то $t_1(\tilde{a}) \vee t_2(\tilde{a}) = 1$ и, следовательно, $D_i(\tilde{a}) = 1$. Если $b_{m-3} = 1$, то $t_{m-1}(\tilde{a}) \vee t_m(\tilde{a}) = 1$ и, значит, вновь $D_i(\tilde{a}) = 1$.

Пусть $b_1 = 1$ и $b_{m-3} = 0$. Тогда найдется такое k, что $b_k = 1$ и $b_{k+1} = 0$. Так как должно быть $\bar{b}_k \vee t_{k+2}(\tilde{a}) \vee b_{k+1} = 1$, то получаем, что $t_{k+2}(\tilde{a}) = 1$. Откуда $D_i(\tilde{a}) = 1$.

Обратно, пусть $D_i(\tilde{a}) = 1$. Тогда существует такое k, что $t_k(\tilde{a}) = 1$. Если $k = 1$ или $k = 2$, то выбираем $b_1 = b_2 = \ldots = b_{m-3} = 0$ и получаем

$F_i(\tilde{a}, \tilde{b}) = 1$. Если $k = m - 1$ или $k = m$, то выбираем $b_1 = b_2 = \ldots = b_{m-3} = 1$ и вновь приходим к равенству $F_i(\tilde{a}, \tilde{b}) = 1$. В остальных случаях для достижения равенства $F_i(\tilde{a}, \tilde{b}) = 1$ полагаем $b_1 = b_2 = \ldots = b_{k-2} = 1$ и $b_k = \ldots = b_{m-3} = 0$.

При построении 3-КНФ F_i для различных i используем различные наборы переменных $\tilde{y}$. В результате описанных построений КНФ K за полиномиальное число шагов будет преобразована в 3-КНФ $F_1 \& \ldots \& F_s$, которая и будет удовлетворять требованиям теоремы.

По аналогии с 3-КНФ и 3-ВЫПОЛНИМОСТЬЮ вводим понятия 2-КНФ и 2-ВЫПОЛНИМОСТИ.

Теорема 4.5. *Задача* 2-ВЫП *полиномиально разрешима, т.е. принадлежит классу* P.

Доказательство. Назовем *резольвентой* дизъюнктов $x_i \vee t_1$ и $\bar{x}_i \vee t_2$ дизъюнкт $D = t_1 \vee t_2$ (в случае отсутствия одного из литералов t_1, t_2 дизъюнкт D совпадает со вторым литералом; в случае отсутствия обоих литералов t_1, t_2 дизъюнкт D считается пустым).

Легко проверяется, что для любых формул A, B справедливо равенство

$$(x_i \vee A) \,\&\, (\bar{x}_i \vee B) = (x_i \vee A) \,\&\, (\bar{x}_i \vee B) \,\&\, (A \vee B).$$

Таким образом, добавление к 2-КНФ резольвенты любой пары дизъюнктов не меняет функцию, реализуемую данной 2-КНФ.

Пусть задана 2-КНФ K. Будем просматривать в K все пары дизъюнктов и добавлять (конъюнктивно) к K их резольвенты. Эту процедуру будем повторять до тех пор, пока не перестанут появляться новые дизъюнкты. Если при этом будет порожден пустой дизъюнкт (который появляется только из пары дизъюнктов вида x_i и $\bar{x}_i$), то, очевидно, исходная 2-КНФ K невыполнима. В противном случае, как мы установим ниже, она выполнима.

Оценим прежде всего трудоемкость предложенного алгоритма. Если длина исходной 2-КНФ равна n, то она содержит не более n переменных, из которых можно построить не более $(2n + 1)^2$ дизъюнктов с одной или двумя переменными. Поэтому порождение новых резольвент будет происходить не более $(2n+1)^2$ раз. При этом число просматриваемых пар дизъюнктов не превосходит $(2n + 1)^4$. Следовательно, описанный выше алгоритм полиномиален.

Итак, предположим, что 2-КНФ K не содержит пустых дизъюнктов и замкнута относительно операции взятия резольвент. Покажем, что в

этом случае она выполнима. Доказательство проведем по числу n переменных, входящих в K.

Базис индукции: $n = 1$. Тогда $K = x_i$ или $K = \bar{x}_i$, и утверждение тривиально верно.

Индуктивный переход. Пусть утверждение верно для всех 2-КНФ с $n < m$ переменными и пусть 2-КНФ K содержит m переменных. Представим K в виде (для упрощения записи знак & опускаем)

$$K = (x_m \vee t_1)(x_m \vee t_2) \dots (x_m \vee t_k)(\bar{x}_m \vee t'_1)(\bar{x}_m \vee t'_2) \dots (\bar{x}_m \vee t'_l) \cdot C_1 \cdot C_2 \cdot \dots \cdot C_r,$$

где t_i, t'_j — литералы либо 0, а $C_1 \cdot C_2 \cdot \dots \cdot C_r$ — 2-КНФ от переменных $x_1, \dots, x_{m-1}$, замкнутая относительно операции взятия резольвент. По предположению индукции существует набор $\tilde{a} = (a_1, \dots, a_{m-1})$, на котором КНФ $C_1 \cdot \dots \cdot C_r$ равна 1. Если существуют такие литералы t_i, t'_j, что $t_i(\tilde{a}) = t'_j(\tilde{a}) = 0$, то также $t_i(\tilde{a}) \vee t'_j(\tilde{a}) = 0$. Однако $t_i \vee t'_j$ — резольвента дизъюнктов $x_m \vee t_i$ и $\bar{x}_m \vee t'_j$. Поэтому дизъюнкт $t_i \vee t'_j$ содержится среди дизъюнктов $C_1, \dots, C_r$. Получаем, что для некоторого v выполняется равенство $C_v(\tilde{a}) = 0$, что невозможно.

Таким образом, либо $t_1(\tilde{a}) = \dots = t_k(\tilde{a}) = 1$, либо $t'_1(\tilde{a}) = \dots = t'_l(\tilde{a}) = 1$. В первом случае КНФ K выполнима на наборе $(\tilde{a}, 0)$, во втором — на наборе $(\tilde{a}, 1)$. Теорема доказана.

УПРАЖНЕНИЯ

16. Доказать NP-полноту следующих задач:

1) задача КЛИКА;

2) задача ВЕРШИННОЕ ПОКРЫТИЕ (см. задачу 15.1).

17. Доказать NP-полноту следующих задач:

1) существование двух противоположных наборов, на которых ДНФ обращается в нуль;

2) распознавание нелинейности булевой функции, заданной в виде ДНФ;

3) распознавание немонотонности булевой функции, заданной в виде ДНФ;

4) распознавание несамодвойственности булевой функции, заданной в виде ДНФ.

§ 8. Примитивно рекурсивные функции

В § 4 мы определили класс $F_{\text{чр}}$ частично рекурсивных функций. В теории рекурсивных функций важную роль играет существенно более простая часть класса $F_{\text{чр}}$ — класс $F_{\text{пр}}$ *примитивно рекурсивных функций.* Примитивно рекурсивные функции — это функции, которые можно получить из исходных функций (4.7) с помощью операций суперпозиции (4.3) и примитивной рекурсии (4.5).

Из определения сразу следует, что всякая примитивно рекурсивная функция всюду определена. Рассмотрим некоторые простые примеры примитивно рекурсивных функций. Функция-константа 0 является исходной примитивно рекурсивной функцией. С помощью суперпозиции функции 0 и исходной функции $x+1$ можно получить любую другую функцию-константу. Отметим, что функции-константы можно считать зависящими от произвольного числа переменных — достаточно воспользоваться исходными селекторными функциями $I_i^n(x_1,\ldots,x_n)$.

Пусть далее функция $\mathrm{sum}(x_1,x_2)$ определяется следующей примитивной рекурсией:

$$\begin{cases} \mathrm{sum}(x_1,0)=x_1, \\ \mathrm{sum}(x_1,x_2+1)=\mathrm{sum}(x_1,x_2)+1 \end{cases} \tag{4.17}$$

(в первом равенстве мы поставили переменную x_1 вместо функции $I_1^1(x_1)$, а во втором — привели только существенную часть определения, опустив те вхождения переменных x_1, x_2, которые можно ввести с помощью функции I_3^3). Нетрудно понять, что примитивная рекурсия (4.17) определяет функцию x_1+x_2.

Подобным образом, примитивные рекурсии

$$\begin{cases} \mathrm{prod}(x_1,0)=0, \\ \mathrm{prod}(x_1,x_2+1)=\mathrm{prod}(x_1,x_2)+x_1, \end{cases}$$

$$\begin{cases} \mathrm{pow}(x_1,0)=1, \\ \mathrm{pow}(x_1,x_2+1)=\mathrm{pow}(x_1,x_2)\cdot x_1 \end{cases}$$

определяют функции $x_1\cdot x_2$ и $x_1^{x_2}$, используя в качестве уже известных функции x_1+x_2 и $x_1\cdot x_2$ (чтобы не делать исключений в определении функции $x_1^{x_2}$, мы приняли $0^0=1$).

Нам бы хотелось доказать примитивную рекурсивность функции $x-y$. Однако это сделать невозможно, поскольку функция $x-y$ принимает

отрицательные значения. Вместо функции $x-y$ мы рассмотрим близкую к ней функцию $x \dot{-} y$, которую определим соотношениями

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

Доказательство примитивной рекурсивности функции $x \dot{-} y$ проведем в два этапа. Сначала установим примитивную рекурсивность более простой функции $x \dot{-} 1$:

$$\begin{cases} 0 \dot{-} 1 = 0, \\ (x+1) \dot{-} 1 = x. \end{cases}$$

Затем, используя функцию $x \dot{-} 1$, получим функцию $x \dot{-} y$:

$$\begin{cases} x \dot{-} 0 = x, \\ x \dot{-} (y+1) = (x \dot{-} y) \dot{-} 1. \end{cases}$$

Суперпозициями функций $x + y,\ x \dot{-} y$ можно определить известные арифметические функции:

$$\min(x, y) = x \dot{-} (x \dot{-} y),$$

$$\max(x, y) = (x + y) \dot{-} \min(x, y),$$

$$|x - y| = (x \dot{-} y) + (y \dot{-} x).$$

Напомним еще о двух важных функциях $\operatorname{sg} x$ и $\overline{\operatorname{sg}}\, x$, которые часто встречаются в теории рекурсивных функций:

$$\operatorname{sg} x = \begin{cases} 0, & \text{если } x = 0, \\ 1, & \text{если } x \neq 0, \end{cases} \qquad \overline{\operatorname{sg}}\, x = \begin{cases} 1, & \text{если } x = 0, \\ 0, & \text{если } x \neq 0. \end{cases}$$

Примитивная рекурсивность функций $\operatorname{sg} x$, $\overline{\operatorname{sg}}\, x$ следует из равенств

$$\overline{\operatorname{sg}}\, x = 1 \dot{-} x, \quad \operatorname{sg} x = \overline{\operatorname{sg}}\, \overline{\operatorname{sg}}\, x.$$

На основе имеющихся примитивно рекурсивных функций можно определить другие интересные примитивно рекурсивные функции. Так, суперпозициями функций (4.7) и функций $x+y,\ x \cdot y$ можно получить любой полином с целыми неотрицательными коэффициентами. Еще более широкие возможности открывает применение функций $x \dot{-} y,\ \operatorname{sg} x,\ \overline{\operatorname{sg}}\, x$. Продемонстрируем одну из этих возможностей.

Пусть $a \in N$. Примитивно рекурсивная функция $\overline{\operatorname{sg}}\, |x - a|$ принимает значение 1 при $x = a$ и значение 0 в остальных случаях. Если

$a_1, \ldots, a_m, b_1, \ldots, b_m$ — числа из N, причем числа $a_1, \ldots, a_m$ попарно различны, то функция

$$b_1 \cdot \overline{\text{sg}}\,|x - a_1| + b_2 \cdot \overline{\text{sg}}\,|x - a_2| + \ldots + b_m \cdot \overline{\text{sg}}\,|x - a_m| \tag{4.18}$$

при $x = a_1, \ldots, a_m$ принимает соответственно значения $b_1, \ldots, b_m$ и равна 0 в остальных случаях.

Идея построения функции (4.18) позволяет «исправлять» значения примитивно рекурсивной функции в конечном числе точек. Именно, рассмотрим, например, примитивно рекурсивную функцию $f(x)$ одной переменной и предположим, что нам необходимо заменить значения функции $f(x)$ в точках $a_1, \ldots, a_m$ значениями $b_1, \ldots, b_m$. Обозначим так «исправленную» функцию $f(x)$ через $f'(x)$. Тогда

$$f'(x) = b_1 \cdot \overline{\text{sg}}\,|x - a_1| + \ldots + b_m \cdot \overline{\text{sg}}\,|x - a_m| + \\ + f(x) \cdot \text{sg}\,|x - a_1| \cdot \ldots \cdot \text{sg}\,|x - a_m|.$$

В предыдущих параграфах при задании функций мы пользовались отношениями (предикатами) вида

$$x = y, \quad x \neq y, \quad x < y, \quad x \leqslant y. \tag{4.19}$$

В принципе можно использовать и более сложные отношения. Мы хотим рассматривать произвольные отношения на множестве N. Чтобы иметь возможность применять отношения при задании функций, удобно ввести характеристические функции отношений. Именно, функцию $f(x_1, \ldots, x_n)$, принимающую лишь значения 0 и 1, называем *характеристической функцией отношения* $\rho(x_1, \ldots, x_n)$, если для любых $x_1, \ldots, x_n$ значение $f(x_1, \ldots, x_n)$ равно 1 в том и только том случае, когда истинно значение $\rho(x_1, \ldots, x_n)$. Отношение ρ считаем примитивно рекурсивным, если примитивно рекурсивна его характеристическая функция.

Для числовых отношений (4.19) характеристическими функциями служат соответственно функции

$$\overline{\text{sg}}\,|x - y|, \quad \text{sg}\,|x - y|, \quad \text{sg}\,(y \dot{-} x), \quad \overline{\text{sg}}\,(x \dot{-} y).$$

Поэтому отношения (4.19) примитивно рекурсивны.

Как мы могли убедиться, к отношениям часто прибегают, когда необходимо определить функцию с помощью так называемого разбора случаев. Сейчас мы рассмотрим подобную ситуацию в общем виде.

Утверждение 4.2. *Пусть функция f определяется схемой*

$$f(x_1,\ldots,x_n)=\begin{cases} f_1(x_1,\ldots,x_n), & \text{если } \rho_1(x_1,\ldots,x_n) \text{ истинно},\\ \ldots\ldots\ldots\ldots & \\ f_m(x_1,\ldots,x_n), & \text{если } \rho_m(x_1,\ldots,x_n) \text{ истинно},\\ f_{m+1}(x_1,\ldots,x_n) & \text{в остальных случаях},\end{cases}$$

где отношения $\rho_1,\ldots,\rho_m$ и функции $f_1,\ldots,f_{m+1}$ примитивно рекурсивны, причем никакие два отношения не являются одновременно истинными. Тогда функция f также примитивно рекурсивна.

Доказательство. Пусть $g_1,\ldots,g_m$ — характеристические функции отношений $\rho_1,\ldots,\rho_m$. Тогда функцию f можно определить равенством

$$\begin{aligned} f(x_1,\ldots,x_n) &= \\ &= f_1(x_1,\ldots,x_n)\cdot g_1(x_1,\ldots,x_n)+\ldots+f_m(x_1,\ldots,x_n)\cdot g_m(x_1,\ldots,x_m)+ \\ &\quad +f_{m+1}(x_1,\ldots,x_n)\cdot \overline{\mathrm{sg}}\,(g_1(x_1,\ldots,x_n)+\ldots+g_m(x_1,\ldots,x_n)). \end{aligned}$$

Из него следует, что функция f примитивно рекурсивна. Утверждение доказано.

В теории рекурсивных функций довольно часто используются операции *ограниченного суммирования*

$$\sum_{i\leqslant x_n} f(x_1,\ldots,x_{n-1},i)$$

и *ограниченного мультиплицирования*

$$\prod_{i\leqslant x_n} f(x_1,\ldots,x_{n-1},i).$$

Покажем, что в случае примитивной рекурсивности функции f функция $g(x_1,\ldots,x_{n-1},x_n)$, полученная из функции f с помощью операции ограниченного суммирования или ограниченного мультиплицирования, также является примитивно рекурсивной. Для операции ограниченного суммирования это следует из соотношений примитивной рекурсии для функции g:

$$\begin{cases} g(x_1,\ldots,x_{n-1},0)=f(x_1,\ldots,x_{n-1},0),\\ g(x_1,\ldots,x_{n-1},x_n+1)=g(x_1,\ldots,x_{n-1},x_n)+f(x_1,\ldots,x_{n-1},x_n+1). \end{cases}$$

Аналогичные построения применимы в случае ограниченного мультиплицирования.

Применение операций ограниченного суммирования и ограниченного мультиплицирования позволяет в ряде случаев значительно упростить доказательство примитивной рекурсивности некоторых функций. Продемонстрируем это на примерах.

Пусть $[x/y]$ есть целая часть от деления x на y, если $y \neq 0$, и есть 0, если $y = 0$ (выбор числа 0 в качестве значения функции $[x/0]$ существенной роли не играет, с равным успехом можно было бы взять любое другое число). Мы хотим установить примитивную рекурсивность функции $[x/y]$. Заметим, что при $y \neq 0$ величина $[x/y]$ есть такое (единственное) число i, что $iy \leqslant x$ и $(i+1)y > x$. Обозначим через $g_1(x, y, i)$ и $g_2(x, y, i)$ характеристические функции отношений $iy \leqslant x$ и $(i+1)y > x$ (они, очевидно, примитивно рекурсивны). Поскольку $[x/y] \leqslant x$, получаем теперь

$$[x/y] = \sum_{i \leqslant x} i \cdot g_1(x, y, i) \cdot g_2(x, y, i).$$

С помощью функции $[x/y]$ легко определить функцию $\mathrm{rm}(x, y)$, равную остатку от деления x на y, если $y \neq 0$, и равную x, если $y = 0$ (вновь значение x при $y = 0$ выбираем лишь для того, чтобы получить более простую формулу для выражения функции $\mathrm{rm}(x, y)$). В самом деле, имеем

$$\mathrm{rm}(x, y) = x \dot{-} [x/y] \cdot y.$$

Найденный выше прием применим для доказательства примитивной рекурсивности еще двух функций $[\sqrt{x}]$ и $[\log_2 x]$ (здесь вновь для простоты положим $\log_2 0 = 0$). Если $g_1(x, i), g_2(x, i)$ — характеристические функции примитивно рекурсивных отношений $i^2 \leqslant x$ и $(i+1)^2 > x$, то

$$[\sqrt{x}] = \sum_{i \leqslant x} i \cdot g_1(x, i) \cdot g_2(x, i).$$

Аналогично, если $h_1(x, i), h_2(x, i)$ — характеристические функции примитивно рекурсивных отношений $2^i \leqslant x$ и $2^{i+1} > x$, то

$$[\log_2 x] = \sum_{i \leqslant x} i \cdot h_1(x, i) \cdot h_2(x, i).$$

Обозначим через $x|y$ отношение «x делит нацело y» (отношение $0|y$ считаем истинным только при $y = 0$). Нетрудно видеть, что отношение $x|y$ эквивалентно отношению

$$[y/x] \cdot x = y,$$

откуда вытекает его примитивная рекурсивность.

Используя отношение $x|y$, подсчитаем количество делителей числа x. Соответствующую функцию обозначим через $d(x)$. Если $g(x,i)$ — характеристическая функция отношения $i|x$, то, очевидно,

$$d(x) = \sum_{i \leqslant x} g(x,i).$$

Обозначим через $\mathrm{Pr}(x)$ отношение «x есть простое число». Тогда $\mathrm{Pr}(x)$ истинно в том и только том случае, когда $x > 1$ и $d(x) = 2$. Следовательно, характеристическая функция отношения $\mathrm{Pr}(x)$ равна произведению характеристических функций отношений $x > 1$ и $d(x) = 2$. Отсюда вытекает примитивная рекурсивность отношения $\mathrm{Pr}(x)$.

Пусть a — простое число и $\exp_a(x)$ равно показателю числа a в разложении x на простые множители (при $x = 0, 1$ значение $\exp_a(x)$ можно выбрать произвольно). Докажем примитивную рекурсивность функции $\exp_a(x)$. Обозначим через $g(x,i)$ характеристическую функцию отношения $a^i|x$. Тогда

$$\exp_a(x) = \sum_{i \leqslant x} g(x, i+1).$$

УПРАЖНЕНИЯ

18. Можно ли в определении примитивно рекурсивной функции отказаться от некоторых функций I_i^n?

19. Доказать, что примитивно рекурсивной является любая периодическая функция $f(x)$ (функция $f(x)$ называется периодическая, если существует такое натуральное число d, что равенство $f(x+d) = f(x)$ выполняется для любого x из N).

20. Пусть $d > 0$. Доказать, что класс примитивно рекурсивных функций замкнут относительно следующего варианта рекурсии по нескольким основаниям:

$$\begin{cases} f(x_1, \ldots, x_{n-1}, 0) = g_0(x_1, \ldots, x_{n-1}), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ f(x_1, \ldots, x_{n-1}, d) = g_d(x_1, \ldots, x_{n-1}), \\ f(x_1, \ldots, x_{n-1}, x_n + d + 1) = \\ \qquad = h(x_1, \ldots, x_{n-1}, x_n, f(x_1, \ldots, x_n), \ldots, f(x_1, \ldots, x_n + d)). \end{cases}$$

21. Пусть $P(x_1, \ldots, x_n), Q(x_1, \ldots, x_n)$ — многочлены с натуральными коэффициентами, а $f(x)$ равно числу решений уравнения

$$P(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n)$$

при ограничениях $0 \leqslant x_1 \leqslant x, \ldots, 0 \leqslant x_n \leqslant x$. Доказать примитивную рекурсивность функции $f(x)$.

22. Пусть $g(x)$ — примитивно рекурсивная функция, а функция $f(x, y)$ определяется соотношениями

$$f(x,y) = \begin{cases} \text{наименьшему решению } z \text{ уравнения } g(z) = y, \\ \text{не превосходящему } x, \text{ если такое решение } z \text{ существует}, \\ 0 \text{ в противном случае.} \end{cases}$$

Доказать, что функция $f(x, y)$ примитивно рекурсивна.

23. Пусть $p(x)$ равно $(x+1)$-у простому числу, т.е. $p(0) = 2$, $p(1) = 3$, $p(2) = 5, \ldots$. Доказать, что функция $p(x)$ примитивно рекурсивна.

24. Пусть $f(x)$ равно $(x+1)$-у десятичному разряду после запятой в разложении числа $\sqrt{2}$. Доказать примитивную рекурсивность функции $f(x)$.

§ 9. Класс частично рекурсивных функций

Приступим к более детальному изучению частично рекурсивных функций. Посмотрим на примерах, как действует операция минимизации. Пусть $g_1(x) = x + 1$ и

$$f_1(x) = (\mu y)(g_1(y) = x).$$

Тогда, как нетрудно видеть,

$$f_1(x) = \begin{cases} x - 1, & \text{если } x > 0, \\ \text{не определено}, & \text{если } x = 0. \end{cases}$$

Пусть $g_2(x)$ есть функция-константа a и

$$f_2(x) = (\mu y)(g_2(y) = x).$$

Тогда

$$f_2(x) = \begin{cases} 0, & \text{если } x = a, \\ \text{не определено}, & \text{если } x \neq a. \end{cases}$$

Интересный результат получается, если операцию минимизации применить к функции $f_1(x)$. Итак, положим

$$f_3(x) = (\mu y)(f_1(y) = x).$$

Тогда, как легко убедиться, функция $f_3(x)$ не будет определена ни при каком значении x. Это так называемая *нигде не определенная функция.* Она играет важную роль в теории алгоритмов. В связи с этим отметим, что применение операции минимизации даже к весьма простым всюду определенным функциям может привести к «существенно» частичным функциям.

Рассмотрим еще несколько примеров применения операции минимизации. Пусть

$$f_4(x) = (\mu y)(y^2 = x), \quad f_5(x) = (\mu y)(2^y = x).$$

Тогда

$$f_4(x) = \begin{cases} \sqrt{x}, & \text{если } x \text{ есть полный квадрат,} \\ \text{не определено} & \text{в противном случае,} \end{cases}$$

$$f_5(x) = \begin{cases} \log_2 x, & \text{если } x \text{ есть степень числа 2,} \\ \text{не определено} & \text{в противном случае.} \end{cases}$$

Применение операции минимизации к одной и той же функции, но по разным переменным, может привести к существенно различным результатам. Так, если

$$f_6(x, y) = (\mu z)(z \dot{-} x = y), \quad f_7(x, y) = (\mu z)(x \dot{-} z = y),$$

то

$$f_6(x, y) = \begin{cases} 0, & \text{если } y = 0, \\ x + y, & \text{если } y \neq 0, \end{cases}$$

$$f_7(x, y) = \begin{cases} x - y, & \text{если } x \geqslant y, \\ \text{не определено}, & \text{если } x < y. \end{cases}$$

Помимо формы (4.6) операция миинимизации применяется еще в нескольких формах. Приведем две из них:

$$f(x_1, \ldots, x_{n-1}) = (\mu y)(g(x_1, \ldots, x_{n-1}, y) = 0),$$

$$f(x_1, \ldots, x_{n-1}) = (\mu y)(g_1(x_1, \ldots, x_{n-1}, y) = g_2(x_1, \ldots, x_{n-1}, y)).$$

Довольно часто операцию минимизации применяют для доказательства частичной рекурсивности функции, заданной алгоритмом, последовательные шаги которого выполняются весьма просто. При этом число шагов алгоритма заранее не может быть ограничено, например, примитивно рекурсивной функцией. Тогда можно поступить так: постулировать существование «длинной» цепочки чисел (которые содержательно

должны кодировать результаты пошаговой работы алгоритма) и затем с помощью простых функций проверить, что данная цепочка действительно удовлетворяет условиям, вытекающим из описания алгоритма. При этом «длинную» цепочку чисел можно будет представить одним число, а существование этого числа возьмет на себя операция минимизации.

В качестве иллюстрации этого приема мы приведем набросок доказательства частичной рекурсивности всюду определенной функции $F(x, y)$, заданной с помощью непримитивной рекурсии. Эта функция не является примитивно рекурсивной, однако доказательство этого факта достаточно громоздко, и мы его не приводим.

Итак, пусть функция $F(x, y)$ определяется следующими равенствами:

$$F(0, y) = y + 2, \quad F(x + 1, 0) = 1,$$
$$F(x + 1, y + 1) = F(x, F(x + 1, y)). \tag{4.20}$$

Убедимся, прежде всего, что равенства (4.20) действительно определяют единственную функцию $F(x, y)$.

В самом деле, если $x = 0$ или $y = 0$, то значения $F(x, y)$ однозначно находятся из первого или второго равенств системы (4.20). Рассмотрим далее значения $F(1, y + 1)$. Пользуясь третьим и первым равенствами, находим, что

$$F(1, y + 1) = F(0, F(1, y)) = F(1, y) + 2.$$

Эти равенства вместе с равенством $F(1, 0) = 1$ дают схему примитивной рекурсии для определения функции $f_1(y) = F(1, y)$:

$$\begin{cases} f_1(0) = 1, \\ f_1(y + 1) = f_1(y) + 2. \end{cases}$$

Понятно, что

$$f_1(y) = F(1, y) = 2y + 1.$$

Если ввести обозначение $f_2(y) = F(2, y)$, то равенства (4.20) вместе с установленным соотношением $F(1, y) = 2y + 1$ приводят к схеме примитивной рекурсии для определения функции $f_2(y)$:

$$\begin{cases} f_2(0) = 1, \\ f_2(y + 1) = 2f_2(y) + 1. \end{cases}$$

Из нее нетрудно получить, что

$$f_2(y) = \underbrace{2(\ldots 2(2}_{y} \cdot 1 + 1) + 1 + \ldots) + 1 = 2^y + 2^{y-1} + \ldots + 2^0 = 2^{y+1} - 1.$$

Итак,

$$F(2,y) = 2^{y+1} - 1.$$

Продолжая в том же духе и вводя обозначение $f_3(y) = F(3,y)$, будем иметь схему примитивной рекурсии для определения функции $f_3(y)$:

$$\begin{cases} f_3(0) = 1, \\ f_3(y+1) = f_2(f_3(y)) = 2^{f_3(y)+1} - 1. \end{cases}$$

Вообще, рассуждая по индукции (обычная индукция по x), можно убедиться в том, что для всякого натурального x функция $f_{x+1}(y) = F(x+1,y)$ (как функция от y) является итерацией функции $f_x(y)$ в точке 1. Эти соображения наводят на мысль, что функция $F(x,y)$ растет слишком быстро, чтобы быть примитивно рекурсивной функцией. Вместе с тем достаточно понятно, что функция $F(x,y)$ алгоримически вычислима — выполнение равенств (4.20) нетрудно запрограммировать на машине Тьюринга.

Итак, обратимся к процессу вычисления функции $F(x,y)$. Пусть $x > 0$ и $y > 0$. Чтобы найти значение $F(x,y)$, нам необходимо согласно равенствам (4.20) иметь определенное количество значений функции F в точках (s,v), где либо $s < x$ и, вообще говоря, $v > y$, либо $s = x$ и $v < y$. Заранее мы не можем сказать, для каких именно точек потребуются иметь значения функции F. Поэтому запишем их пока в виде последовательности с неопределенными величинами $v_0, \ldots, v_{x-1}$:

$$F(0,0), F(0,1), \ldots, F(0,v_0); \quad F(1,0), F(1,1), \ldots, F(1,v_1); \ldots;$$

$$F(x-1,0), F(x-1,1), \ldots, F(x-1,v_{x-1}); \; F(x,0), F(x,1), \ldots, F(x,y-1). \tag{4.21}$$

Часть значений из последовательности (4.21) дается непосредственно равенствами (4.20). Именно

$$F(0,0) = 2, \; F(0,1) = 3, \ldots, F(0,v_0) = v_0 + 2,$$

$$F(1,0) = F(2,0) = \ldots = F(x,0) = 1. \tag{4.22}$$

Что касается остальных значений, то для их получения нам придется воспользоваться третьим из равенств (4.20).

Итак, найти значение $F(x,y)$ можно, если предварительно создать последовательность (4.21) с достаточно большими (и пока неопределенными) величинами $v_0, \ldots, v_{x-1}$, в которой часть значений известна и дается

равенствами (4.22), а остальная часть значений определяется рекурсивно с использованием третьего из равенств (4.20) (предполагается, что необходимые для этого значения $F(x, y-1)$ и $F(x-1, F(x, y-1))$ присутствуют в последовательности (4.21)).

В последовательность (4.21) входит символ функции F. Тем самым мы как бы предполагаем, что нужные нам значения функции F уже найдены. Однако это не так. Поэтому последовательность (4.21) имеет смысл переписать несколько иначе:

$$(0,0,b_{00}),(0,1,b_{01}),\ldots,(0,v_0,b_{0v_0});\ (1,0,b_{10}),(1,1,b_{11}),\ldots,(1,v_1,b_{1v_1});\ldots$$
$$\ldots;(x-1,0,b_{x-1,0}),(x-1,1,b_{x-1,1}),\ldots,(x-1,v_{x-1},b_{x-1,v_{x-1}});$$
$$(x,0,b_{x0}),(x,1,b_{x1}),\ldots,(x,y-1,b_{x,y-1}). \qquad (4.23)$$

В этой последовательности числа $b_{00}, b_{01}, \ldots, b_{0v_0}$ и $b_{10}, \ldots, b_{x-1,0}$ определяются согласно первым двум равенствам (4.20), а остальные числа — с использованием третьего равенства (4.20). Наборы из троек чисел, составляющие последовательность (4.23), мы закодируем далее числами из N (о способе кодирования чуть позже). В результате образуется последовательность кодов

$$c_{00}, c_{01}, \ldots, c_{0v_0};\ c_{10}, c_{11}, \ldots, c_{1v_1}; \ldots; c_{x-1,0}, c_{x-1,1}, \ldots, c_{x-1,v_{x-1}};$$
$$c_{x0}, c_{x1}, \ldots, c_{x,y-1}. \qquad (4.24)$$

Наконец, эту последовательность также закодируем одним числом d (способ кодирования пока оставляем в стороне).

Теперь для вычисления значения $F(x, y)$ следует найти наименьшее число d, которое есть код последовательности вида (4.24), в которой числа c_{ij} в свою очередь являются кодами троек (i, j, b_{ij}). Данные тройки должны образовывать последовательность (4.23), элементы которой подчиняются известным нам соотношениям.

Можно показать, что свойство числа d, которое изложено в предыдущем абзаце, выразимо с помощью примитивно рекурсивной функции. Разумеется, на этом пути есть определенные технические трудности (но мы не останавливаемся на их разрешении, поскольку в следующем параграфе будет изложено решение более общей задачи). Тем не менее предложенный путь доказательства частичной рекурсивности функции $F(x, y)$ представляется одним из наиболее доступных.

Обратимся теперь к вопросу о кодировании троек. Сначала определим примитивно рекурсивную функцию, кодирующую пары. В качестве

такой функции возьмем функцию

$$c(x,y) = (x+y)^2 + x.$$

Несложно убедиться, что функция c инъективна. Вместе с тем функция c принимает далеко не все значения из N. Именно, при любых x, y в ее область значений не входят числа

$$(x+y)^2 + x + 1, \ldots, (x+y)^2 + 2x + 2y.$$

Тем не менее по коду $v = c(x,y)$ пары (x,y) довольно просто вычислить элементы x и y. Это достигается с помощью примитивно рекурсивных функций $l(v)$ и $r(v)$:

$$l(v) = v \dot{-} [\sqrt{v}]^2, \quad r(v) = [\sqrt{v}] \dot{-} l(v).$$

Кодирование троек можно осуществить с помощью функции

$$c^3(x,y,z) = c(c(x,y),z).$$

«Обратными» функциями, доставляющими по коду $v = c^3(x,y,z)$ компоненты x, y, z, будут суперпозиции функций l и r: соответственно $l(l(v))$, $r(l(v))$ и $r(v)$.

Вообще, при любом $n \geqslant 3$ кодирование n-ок можно выполнить с помощью функции $c^n(x_1, \ldots, x_n)$, где

$$c^2(x_1,x_2) = c(x_1,x_2),\ c^{n+1}(x_1,\ldots,x_{n+1}) = c(c^n(x_1,\ldots,x_n),x_{n+1}).$$

Если $v = c^n(x_1,\ldots,x_n)$, то

$$x_1 = \underbrace{l(\ldots l}_{n-1}(v)\ldots),\ x_2 = r(\underbrace{l(\ldots l}_{n-2}(v)\ldots)), \ldots, x_{n-1} = r(l(v)),\ x_n = r(v).$$

Для кодирования последовательности (4.24) одним числом d также можно использовать подходящую функцию c^n. Чтобы при декодировании избежать многократных суперпозиций функции l, определим примитивно рекурсивные функции l' и l_1:

$$\begin{cases} l'(v,0) = v, \\ l'(v,i+1) = l(l'(v,i)), \end{cases} \qquad l_1(v,i) = r(l'(v,i)).$$

Если теперь $v = c^{n+1}(a_n,\ldots,a_0)$, то при любом $i < n$ будем иметь $l_1(v,i) = a_i$.

УПРАЖНЕНИЯ

25. Доказать, что результат применения операции минимизации к всюду определенной функции есть функция, определенная хотя бы в одной точке.

26. Применить операцию минимизации к функциям $x+y$, $x\cdot y, x-y$ (значение этой функции не определено, если $x<y$) и x/y (значение этой функции считаем неопределенным, если либо $y=0$, либо $y\neq 0$ и x не является кратным y).

27. Подобрать такие функции $g_1(x,y), g_2(x,y)$, чтобы применение к ним операции минимизации давало функции $x+y$ и $x\cdot y$.

28. Пусть $a_1,\ldots,a_s$ — различные числа из N и

$$f(x)=\begin{cases} 1, & \text{если } x\in\{a_1,\ldots,a_s\},\\ \text{не определено} & \text{в противном случае.}\end{cases}$$

Доказать, что функция $f(x)$ частично рекурсивна.

29. Доказать частичную рекурсивность функции $g(x)$, если

$$g(x)=\begin{cases} 0, & \text{если существует такое } i, \text{ что } l_1(x,i)=1,\\ \text{не определено} & \text{в противном случае.}\end{cases}$$

30. Довести до конца доказательство частичной рекурсивности функции $F(x,y)$.

31. Разработать такую примитивно рекурсивную нумерацию пар, чтобы каждое число из N было номером ровно одной пары.

§ 10. Частичная рекурсивность вычислимых функций. Формула Клини

В § 4 мы доказали, что всякая частично рекурсивная функция вычислима на машине Тьюринга. В этом параграфе мы установим, что $\mathrm{F}_{\text{выч}}\subseteq\mathrm{F}_{\text{чр}}$.

⌋ Начнем с того, что определим примитивно рекурсивную функцию θ_n, которая позволяет нумеровать основной код n-ки чисел на ленте машины Тьюринга. Итак, пусть $\theta_n(x_1,\ldots,x_n)$ есть число, двоичная запись которого представляет собой основной код набора $(x_1,\ldots,x_n)$ (крайний левый и крайний правый символы этой записи суть 1). Для функции θ_1 имеем следующую примитивную рекурсию:

$$\begin{cases}\theta_1(0)=1,\\ \theta_1(x+1)=2\theta_1(x)+1.\end{cases}$$

Функция θ_n определяется через функцию θ_{n-1} согласно следующей примитивной рекурсии:

$$\begin{cases} \theta_n(x_1,\ldots,x_{n-1},0) = 4\theta_{n-1}(x_1,\ldots,x_{n-1}) + 1, \\ \theta_n(x_1,\ldots,x_{n-1},x_n+1) = 2\theta_n(x_1,\ldots,x_n) + 1. \end{cases}$$

Теорема 4.6 (формула Клини) *Для любой вычислимой (на машине Тьюринга) функции $f(x_1,\ldots,x_n)$ найдутся такие примитивно рекурсивные функции $G(x_1,\ldots,x_n,y)$ и $H_f(x_1,\ldots,x_n,y)$, что имеет место представление*

$$f(x_1,\ldots,x_n) = G(x_1,\ldots,x_n,(\mu y)(H_f(x_1,\ldots,x_n,y)=0)).$$

Доказательство. Пусть машина Тьюринга $\mathcal{M}$ правильно вычисляет функцию f. Будем предполагать, что программа машины $\mathcal{M}$ преобразована таким образом, что при попадании в заключительное состояние q_0 машина $\mathcal{M}$ остается в этом состоянии, не сдвигает головку на ленте и не меняет символа, обозреваемого головкой (мы так уже поступали при доказательстве теоремы Кука в § 7).

Положим $x = (x_1,\ldots,x_n)$. Рассмотрим произвольный момент времени t в вычислении значения $f(x)$ на машине $\mathcal{M}$. Пусть машина $\mathcal{M}$ в этот момент находится в состоянии q_i, слева от головки располагается часть ленты L_t, справа — часть R_t (головка находится в первой клетке части R_t). Обозначим через $l(x,t)$ число, двоичная запись которого представлена в части L_t (если в L_t нет единиц, то полагаем $l(x,t)=0$). Аналогичное обозначение $r(x,t)$ вводим для числа, представленного в части R_t, однако здесь по техническим причинам нам удобно читать двоичную запись слева направо (младший разряд двоичной записи числа $r(x,t)$ расположен в самой левой клетке части R_t). Положим также $q(x,t)=i$.

В момент времени $t=0$ будем иметь

$$l(x,0)=0, \quad r(x,0)=\theta_n(x_n,\ldots,x_1), \quad q(x,0)=1.$$

Тройку чисел $l(x,t), r(x,t), q(x,t)$ запишем в виде одного числа

$$\mathrm{Code}(x,t) = c^3(l(x,t), r(x,t), q(x,t)),$$

которое будем считать кодом конфигурации машины $\mathcal{M}$ в момент времени t.

Обозначим через $\rho(x)$ функцию, которая для числа x вида 2^y-1, где $y>0$, равна $y-1$. Тогда функцию f можно представить в виде

$$f(x_1,\ldots,x_n) = \rho(r(l(\mathrm{Code}(x,(\mu y)(r(\mathrm{Code}(x,t))=0))))).$$

Из этой формулы следует, что теорема будет доказана, если мы установим примитивную рекурсивность функций Code и ρ.

В качестве функции $\rho(x)$ можно взять функцию $[\log_2(x+1)] \dot{-} 1$. Функцию Code определим примитивной рекурсией по t. Имеем

$$\mathrm{Code}(x,t) = c^3(0, \theta_n(x_n \ldots, x_1), 1).$$

Чтобы выразить значение $\mathrm{Code}(x,t+1)$ через значение $\mathrm{Code}(x,t)$, определим сначала величины $l(x,t), r(x,t)$ и $q(x,t)$:

$$l(x,t) = l(l(\mathrm{Code}(x,t))),\ r(x,t) = r(l(\mathrm{Code}(x,t))),\ q(x,t) = l(\mathrm{Code}(x,t)).$$

Затем найдем значение первого разряда в двоичном представлении числа $r(x,t)$. Это будет

$$\nu(x,t) = \mathrm{rm}(r(x,t), 2).$$

Если

$$a = \nu(x,t), \quad i = q(x,t),$$

то находим в программе машины $\mathcal{M}$ команду

$$aq_i \to bDq_j,$$

где $D \in \{L, R, S\}$. Далее в соответствии со значениями b, D, j определяем значения $l(x,t+1), r(x,t+1)$ и $q(x,t+1)$. Очевидно, что $q(x,t+1) = j$. Если $D = S$, то $l(x,t+1) = l(x,t)$ и

$$r(x,t+1) = (r(x,t) \dot{-} \nu(x,t)) + b.$$

Пусть $D = L$. Тогда

$l(x,t+1) = \left[\frac{l(x,t)}{2}\right]$,

$r(x,t+1) = 2((r(x,t) \dot{-} \nu(x,t)) + b) + \mathrm{rm}(l(x,t), 2)$.

Аналогично, если $D = R$, то

$l(x,t+1) = 2l(x,t) + b$,

$r(x,t+1) = \left[\frac{r(x,t)}{2}\right]$.

Таким образом, величину $\mathrm{Code}(x,t+1)$ можно определить по формуле

$$\mathrm{Code}(x,t+1) = \sum \overline{\mathrm{sg}}|a - \nu(x,t)| \cdot \overline{\mathrm{sg}}|i - l(\mathrm{Code}(x,t))| \cdot \\ \cdot c^3(l(x,t+1), r(x,t+1), q(x,t+1)),$$

где (конечная) сумма распространяется по всем наборам (a, i), а величины $l(x, t+1), r(x, t+1)$ и $q(x, t+1)$ определяются, как указано выше. Теорема доказана.

Из теорем 4.1 и 4.6 вытекает важное следствие.

Теорема 4.7. *Классы* $\mathrm{F}_{\text{выч}}$ *и* $\mathrm{F}_{\text{чр}}$ *совпадают.*

УПРАЖНЕНИЯ

32. Используя идею доказательства теоремы 10.6, показать, что характеристические функции (арифметических) отношений из класса P являются примитивно рекурсивными. Получить аналогичный результат для отношений класса NP.

33. С использованием теоремы 10.6 доказать, что любое непустое множество чисел, перечислимое (возможно, с повторениями) частично рекурсивной функцией, перечислимо также подходящей примитивно рекурсивной функцией.

Ответы, решения, указания

Глава 1.

1. Ответ: $\sum_{i=0}^{n}(-1)^i\binom{n}{i}k^{k^{n-i}}$. Используется принцип включения-исключения. Первое слагаемое суммы (полагаем $(-1)^0 = 1$) есть k^{k^n} — число всех функций в $P_k^{(n)}$. Из него вычитается n чисел вида $k^{k^{n-1}}$, каждое из них есть количество функций из $P_k^{(n)}$, не зависящих существенно от фиксированной переменной. Поскольку эти множества пересекаются, далее необходимо добавить $\binom{n}{2}$ слагаемых вида $k^{k^{n-2}}$, которые «отвечают» за множества функций, не зависящих существенно от фиксированных двух переменных, и т.д.

2. Имеем $k-1 = J_1(1)$, $0 = J_1(k-1)$. Функция $f_1(x) = \max(1, x, J_1(x), \ldots, J_{k-2}(x))$ равна 1 при $x = 0$ и равна $k-1$ при $x \neq 0$. Поэтому $J_0(x) = J_1(f_1(x))$. Аналогично определяем $f_2(x) = \max(J_0(x), \ldots, J_{k-2}(x))$ и получаем $J_{k-1}(x) = J_0(f_2(x))$.

3. Имеем $0 = x \dot{-} x$, $\min(x, y) = x \dot{-} (x \dot{-} y)$. Из константы 0 с помощью функции $\bar{x}$ получаем остальные константы. Далее, $\sim x = (k-1) \dot{-} x$, $\max(x, y) = \sim \min(\sim x, \sim y)$. Функция $f(x) = 1 \dot{-} (1 \dot{-} x)$ равна 0 при $x = 0$ и равна 1 при $x \neq 0$. Поэтому функцию $J_0(x)$ можно получить последовательным вычитанием (посредством функции $\dot{-}$) $k-1$ раз из константы $k-1$ функции $f(x)$. Теперь при $i \neq 0$ получаем $J_i(x) = J_0(x+k-i)$.

4. Имеем $0 = \min(x, x+1, \ldots, x+k-1)$. С помощью функции $\bar{x}$ из константы 0 получаем остальные константы. Функция $f_0(x) = \min(x+1, \ldots, x+k-1) + k - 1$ равна 0 при $x = 0$ и равна $k-1$ при $x \neq 0$. При $1 \leqslant i \leqslant k-1$ образуем функции $f_i(x) = f_0(x+k-i)$, обладающие аналогичными свойствами. Поэтому при любом $l \in E_k$ будем иметь $J_l(x) = \min_{i \neq l}(f_i(x))$. При $l \neq 0$ функция $f_{i,l}(x) = \min(l, f_i(x))$ равна 0 при $x = i$ и равна l при $x \neq i$, а функция $g_{i,l,s}(x) = f_{i,l}(x) + s$ — соответственно s и $l + s \pmod{k}$. Выбирая $l = k - 1 - s$, получаем функцию $h_{i,s}(x)$, которая равна s при $x = i$ и равна $k-1$ при $x \neq i$. Теперь имеем $\sim x = \min(h_{0,k-1}(x), h_{1,k-2}(x), \ldots, h_{k-1,0}(x))$. Наконец, $\max(x, y) = \sim \min(\sim x, \sim y)$.

5. Данную систему сведем к системе $\{j_0(x), x+y\}$. Заметим, что сумма $k-1$ слагаемых $J_i(x)$ дает функцию $j_i(x)$. Из функций 1 и $x+y$ получаем функцию $x+i$. При $i \neq 0$ имеем $j_0(x) = j_i(x+i)$.

6. Имеем $(k-1)^2 = 1$. Функция $f_{i,s}(x) = s \cdot (k-1) \cdot J_i(x)$ равна s при $x = i$ и равна 0 при $x \neq i$. Из функций $f_{i,s}$ с помощью функции max можно «собрать» любую одноместную функцию, в частности, функцию $\sim x$.

7. Покажем сначала, как можно получить любую функцию $f(x_1, \ldots, x_n)$ из P_k, которая принимает лишь значения 0,1. Функция $j_{i_1}(x_1) \cdot \ldots \cdot j_{i_n}(x_n)$ (получаемая суперпозициями функций заданной системы) принимает значение 1 на наборе $(i_1, \ldots, i_n)$ и значение 0 на остальных наборах. Функция $d(x_1, x_2) = j_0(j_0(x_1) \cdot j_0(x_2))$ на множестве E_2^2 совпадает с дизъюнкцией. Поэтому функцию f можно «собрать» из функций вида $j_{i_1}(x_1) \cdot \ldots \cdot j_{i_n}(x_n)$ с помощью функции d. Предположим далее, что функция f принимает лишь значения из E_3. Определяем функции $f_1(x_1, \ldots, x_n) = \bar{j}_0(f(x_1, \ldots, x_n))$ и $f_2(x_1, \ldots, x_n)$, которая равна 1, если f равна 2, и равна 0 в остальных случаях. Тогда функция f представима в виде произведения функций $f_1(x_1, \ldots, x_n)$ и $f_2(x_1, \ldots, x_n) + 1$. Этот процесс повторяем в случае, когда функция f принимает лишь значения из множества E_4, и т.д.

8. Каждая из двуместных функций вместе с множеством $P_k^{(1)}$ образует полную в P_k систему. Поэтому алгоритмом достаточно порождать лишь все одноместные функции из замыкания исследуемой системы.

9. Выпишем все множества G_i для случая $k = 2$: $\{e_1^2, e_2^2\}$ (далее в целях сокращения записи функции e_1^2, e_2^2 в множествах G_i не приводим), $\{0\}$, $\{1\}$, $\{0,1\}$, $\{\bar{e}_1^2, \bar{e}_2^2\}$, $\{0,1,\bar{e}_1^2,\bar{e}_2^2\}$, $\{x \vee y\}$, $\{xy\}$, $\{x \vee y, xy\}$, $\{0, x \vee y\}$, $\{0, xy\}$, $\{0, x \vee y, xy\}$, $\{0, x \oplus y\}$, $\{0, xy, x\bar{y}, \bar{x}y\}$, $\{0, x \vee y, xy, x \oplus y, x\bar{y}, \bar{x}y\}$,

$\{1, x \vee y\}$, $\{1, xy\}$, $\{1, x \vee y, xy\}$, $\{1, x \oplus y \oplus 1\}$, $\{1, x \vee y, x \vee \bar{y}, \bar{x} \vee y\}$, $\{1, x \vee y, xy, x \oplus y \oplus 1, x \vee \bar{y}, \bar{x} \vee y\}$, $\{0, 1, x \vee y\}$, $\{0, 1, xy\}$, $\{0, 1, x \vee y, xy\}$, $\{0, 1, \bar{e}_1^2, \bar{e}_2^2, x \oplus y, x \oplus y \oplus 1\}$. Довольно легко убедиться, что множества $\{\bar{e}_1^2, \bar{e}_2^2\}$, $\{0, x \vee y, xy, x \oplus y, x\bar{y}, \bar{x}y\}$, $\{1, x \vee y, xy, x \oplus y \oplus 1, x \vee \bar{y}, \bar{x} \vee y\}$, $\{0, 1, x \vee y, xy\}$, $\{0, 1, \bar{e}_1^2, \bar{e}_2^2, x \oplus y, x \oplus y \oplus 1\}$ определяют соответственно известные классы S, T_0, T_1, M, L. Оставшаяся часть доказательства — определение всех остальных классов F_i' и проверка их на включение — сравнительно трудная задача. Например, можно проверить, что множество $\{0, 1, \bar{e}_1^2, \bar{e}_2^2\}$ определяет класс всех функций, существенно зависящих не более чем от одной переменный (если функция существенно зависит более чем от двух переменных, то из нее подстановкой констант 0,1 и функций e_1^2, e_2^2 можно получить функцию, существенно зависящую ровно от двух переменных). Этот класс, конечно, целиком содержится в классе L линейных функций. Множество $\{0, x \vee y, xy\}$ определяет (замкнутый) класс всех монотонных функций, сохраняющих константу 0, т.е. класс $M \cap T_0$. Множество $\{0, 1, x \vee y\}$ определяет класс всех дизъюнкций, т.е. класс всех функций, имеющих вид $a_0 \vee a_1 x_1 \vee \ldots \vee a_n x_n$, где коэффициенты $a_0, a_1, \ldots, a_n$ независимым образом принимают значения из E_2. Этот класс, разумееется, целиком содержится в классе M монотонных функций.

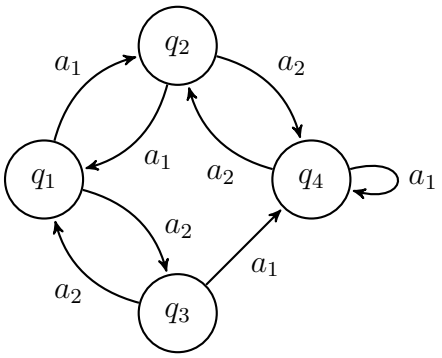
10. Указание: всякий замкнутый класс из F, содержащий бесконечное число функций f_n, совпадает с классом F.

11. Следует объединить конструкции из теорем 1.6 и 1.7. Определим последовательность функций $\{f_0, f_1, \ldots\}$ аналогично последовательности из теоремы 1.6, но с заменой 1 на 3. Если $\{n_1, n_2, \ldots\}$ — последовательность чисел, больших 1, то определим F как замыкание системы функций, состоящей из функций $f_0, f_1, \ldots$ и $g_{n_1}, g_{n_2}, \ldots$. Нетрудно показать, что класс F состоит из функций $f_0, f_1, \ldots$ и функций, образованных суперпозициями функций $g_{n_1}, g_{n_2}, \ldots$. Предположим, что класс F имеет базис B. Устанавливаем, что базис B содержит хотя бы одну функцию f_n, где $n \neq 0$. Так же, как в теореме 1.6, убеждаемся, что в базис B не могут входить две различные функции f_m, f_n, где $m, n \neq 0$. Если в базис B входит только одна функция f_m, где $m \neq 0$, то, как нетрудно показать, из функций базиса B невозможно получить функцию f_n, где $n > m$. Таким образом, класс F не имеет базиса. Чтобы завершить решение задачи, остается показать, что при $n \notin \{n_1, n_2, \ldots\}$ функция g_n не входит в класс F. В самом деле, функция g_n не может быть реализована формулой вида $f_m(\Phi_1, \ldots, \Phi_m)$, поскольку функция f_m не принимает

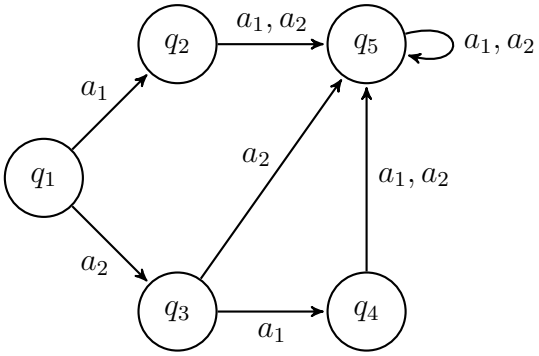
значение 1. То, что функция g_n не может быть реализована формулой вида $g_{n_i}(\Phi_1, \ldots, \Phi_{n_i})$, доказывается так же, как в теореме 1.7.

Глава 2.

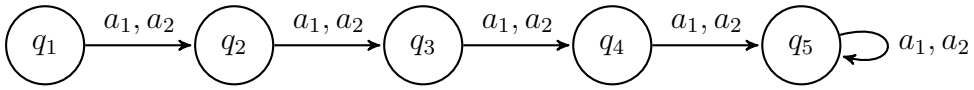
1.



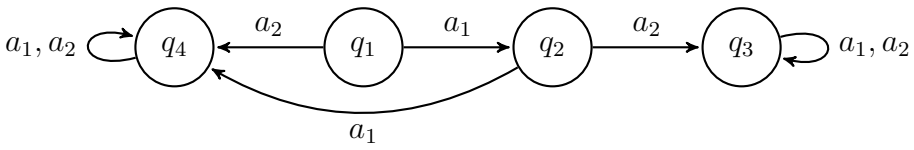
2. В пп. a)–d) множество F есть соответственно $\{q_2, q_4\}, \{q_4\}, \{q_3\}, \{q_3\}$.



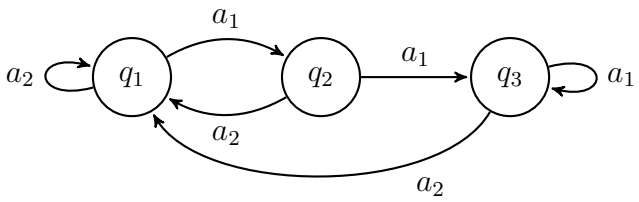
a)



b)



c)



d)

e) По аналогии с пп. c) и d).

3. a) Для произвольных слов $\bar{a}, \bar{b}$ в алфавите $\{a_1, a_2\}$ полагаем $\bar{a} \sim \bar{b}$ в том и только том случае, когда либо $\bar{a} = \bar{b} = \Lambda$, либо $\bar{a} \neq \Lambda$ и $\bar{b} \neq \Lambda$. Тогда множество всех слов в алфавите $\{a_1, a_2\}$ разбивается на два класса эквивалентности, один из которых есть $\{\Lambda\}$.
b) Для произвольных слов $\bar{a}, \bar{b}$ полагаем $\bar{a} \sim \bar{b}$ в том и только том случае, когда либо $\bar{a} = \bar{b} = \Lambda$, либо $\bar{a} = \bar{b} = a_1$, либо $\bar{a}, \bar{b}$ — слова, отличные от Λ и a_1. Множество всех слов в алфавите $\{a_1, a_2\}$ при этом разбивается на три класса эквивалентности, один из которых есть $\{a_1\}$.
c) Для произвольных слов $\bar{a}, \bar{b}$ полагаем $\bar{a} \sim \bar{b}$ в том и только том случае, когда либо $\bar{a} = \bar{b} = \Lambda$, либо $\bar{a}, \bar{b}$ — буквы алфавита $\{a_1, a_2\}$ (возможно, различные), либо $\bar{a}, \bar{b}$ — слова длины, не меньшей двух. Множество всех слов в алфавите $\{a_1, a_2\}$ разбивается на три класса эквивалентности, два из которых суть $\{\Lambda\}$ и $\{a_1, a_2\}$. Их объединение даёт искомое множество $\{\Lambda, a_1, a_2\}$.
d) Для произвольных слов $\bar{a}, \bar{b}$ полагаем $\bar{a} \sim \bar{b}$ в том и только том случае, когда либо $\bar{a}, \bar{b}$ суть слова вида a_1^n (возможно, различные), либо $\bar{a}, \bar{b}$ суть слова вида $a_1^n a_2$ (также допускаются различные значения параметра n), либо $\bar{a}, \bar{b}$ не имеют вида a_1^n или $a_1^n a_2$. Множество всех слов в алфавите $\{a_1, a_2\}$ разбивается на три класса эквивалентнсоти, один из которых есть искомое множество $\{a_1^n a_2 : \ n \geq 0\}$.

4. Проще всего определить эквивалентность так, чтобы классы эквивалентности состояли из всех слов длины соответственно $kn, kn+1, \ldots, kn+n-1$ $(k = 0, 1, \ldots)$.

5. Эту задачу лучше решать с использованием недетерминированных автоматов либо регулярных множеств и теоремы Клини. В последнем случае следует воспользоваться операцией обращения. Нетрудно видеть, для всякой левоинвариантной эквивалентности конечного индекса $\sim_l$ можно определить аналогичную провоинвариантную эквивалентность конечного индекса $\sim_r$, причём произвольные слова $\bar{a}, \bar{b}$ будут эквивалентны в смысле отношения $\sim_l$ в том и только том случае, когда

эквивалентны в смысле отношения $\sim_r$ обращения слов $\bar{a}, \bar{b}$.

6. d) Можно взять три множества: «чётная длина», «начало a_1», «чередование a_1 и a_2».

7. Пусть $\mathcal{A} = (A, Q, f, I, F)$ — источник с множеством начальных состояний I. Если I состоит из нескольких состояний, то $D(\mathcal{A})$ есть объединение всех множеств, допустимых автоматами (A, Q, f, q_j, F), где $q_j \in I$. Поэтому далее можно рассматривать источники, имеющие только одно начальное состояние. Пусть $\mathcal{A}' = (A, Q, f, q_1, F)$ — такой источник. Введём новое состояние q', не принадлежащее множеству Q. Определим недетерминированный автомат $\mathcal{A}'' = (A, Q \cup \{q'\}, f', q_1, F)$, где функция переходов f' получается из функции f следующим образом. Если $a_i \in A, q_j \in Q$ и $f(a_i, q_j)$ определено, то пусть $f'(a_i, q_j) = f(a_i, q_j)$. В противном случае полагаем $f'(a_i, q_j) = q'$. Для всех букв a_i полагаем также $f'(a_i, q') = q'$. Нетрудно видеть, что для недетерминированного автомата $\mathcal{A}''$ справедливо равенство $D(\mathcal{A}'') = D(\mathcal{A})$.

8. $(a_1 \cup a_2)^* \cdot a_2 \cdot a_2 \cdot a_1 \cdot (a_1 \cup a_2)^*$.

9. Один раз. Если слово $\bar{a}$ состоит из n букв, то сначала получаем множество всех слов в алфавите A, имеющих длину больше n:

$$\underbrace{(a_1 \cup \ldots \cup a_k) \cdot \ldots \cdot (a_1 \cup \ldots \cup a_k)}_{n+1} \cdot (a_1 \cup \ldots \cup a_k)^*.$$

Затем к этому множеству с помощью операции объединения добавляем все слова длины, не превосходящей n, отличные от $\bar{a}$.

10. a) Пусть конечное множество состоит из слов $\bar{a}_1, \ldots, \bar{a}_n$ и, например, $\bar{a}_i = a_{i_1} \ldots a_{i_s}$. Тогда регулярное выражение $a_{i_1} \cdot \ldots \cdot a_{i_s}$ (скобки опускаем) определяет регулярное множество, состоящее из одного слова $\bar{a}_i$. Далее с помощью операции объединения образуем регулярное выражение, которое определяет множество $\{\bar{a}_1, \ldots \bar{a}_n\}$.

b) Пусть $X = \{a_1, a_2\}^* \setminus \{\bar{a}_1, \ldots, \bar{a}_n\}$ и m — максимум из длин слов $\bar{a}_1, \ldots, \bar{a}_n$. Сначала образуем регулярное выражение

$$\underbrace{(a_1 \cup a_2) \cdot \ldots \cdot (a_1 \cup a_2)}_{m+1} \cdot (a_1 \cup a_2)^*,$$

которое определяет множество всех слов в алфавите $\{a_1, a_2\}$, имеющих длину больше m. Затем, пользуясь п. a) задачи, строим регулярное выражение, которое определяет множество всех слов длины, не превосходящей m, и отличных от слов $\bar{a}_1, \ldots, a_n$. На последнем этапе с помощью знака $\cup$ соединяем полученные регулярные выражения.

c) Искомое множество определяется регулярным выражением $(\bar{a}_1 \cup \ldots \cup \bar{a}_n)^*$, если в это множество включить пустое слово. В противном случае следует взять регулярное выражение

$$(\bar{a}_1 \cup \ldots \cup \bar{a}_n) \cdot (\bar{a}_1 \cup \ldots \cup \bar{a}_n)^*.$$

d) Если $\bar{a} = \Lambda$, то искомое регулярное выражение есть $(a_1 \cup a_2)^*$. Пусть $\bar{a} \neq \Lambda$ и $\bar{a} = a_{i_1} \ldots a_{i_s}$. Тогда соответствующим регулярным выражением будет выражение

$$(a_1 \cup a_2)^* \cdot a_{i_1} \cdot \ldots \cdot a_{i_s} \cdot (a_1 \cup a_2)^*.$$

e) Будем считать, что слово $\bar{a}$ непусто (в противном случае решением задачи является пустое множество $\varnothing$). Поскольку в общем случае решение задачи довольно громоздко, рассмотрим два частных случая. Пусть $\bar{a} = a_1$. Тогда множество всех слов, не содержащих буквы a_1, — это множество всех слов, состоящих только из буквы a_2 (включая пустое слово). Соответствующее регулярное выражение имеет вид a_2^*. Пусть теперь $\bar{a} = a_1 a_1 a_2$. Тогда произвольное слово, не содержащее подслово $a_1 a_1 a_2$, имеет вид

$$a_2^{k_1} a_1 a_2^{l_1} a_1 a_2^{l_2} \ldots a_1 a_2^{l_p} a_1^{k_2},$$

где k_1, k_2, p — произвольные неотрициательные, а $l_1, \ldots, l_p$ — произвольные натуральные числа. Из этого представления следует вид регулярного выражения, решающего поставленную задачу: $a_2^* \cdot (a_1 \cdot a_2^* \cdot a_2)^* \cdot a_1^*$. Отметим ещё, что для произвольного слова $\bar{a}$ задачу проще решить, если сначала определить детерминированный автомат, который допускает множество всех слов, содержащих в качестве подслова слово $\bar{a}$. Затем следует перейти к дополнению этого множества и воспользоваться теоремой Клини.

11. Необходимо воспользоваться тем же приёмом, что и при доказательстве второй части теоремы Клини. Отличие в доказательстве будет состоять в том, что переход из одного состояния в непосредственно следующее состояние осуществляется под действием слова, которое, вообще говоря, не является буквой входного алфавита.

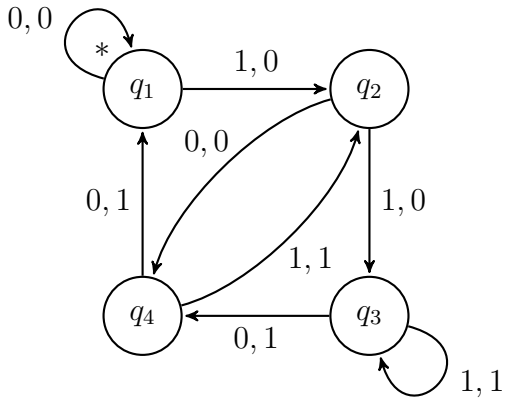
12. Необходимо применить теорему Клини. Если регулярное выражение α над алфавитом A определяет конечно-автоматное множество X, а регулярные выражения $\beta_1, \ldots, \beta_k$ над алфавитом B — конечно-автоматные множества $Y_1, \ldots, Y_k$, то, как нетрудно заметить, множество $X[Y_1, \ldots, Y_k]$ будет определяться регулярным выражением, которое

поолучается из регулярного выражения α заменой символов $a_1, \ldots, a_k$ соответственно регулярными выражениями $\beta_1, \ldots, \beta_k$. Значит, множество $X[Y_1, \ldots, Y_k]$ также конечно-автоматно.

13. Следует воспользоваться регулярными выражениями. Из операций объединения, произведения и итерации на порядок букв в словах влияние оказывает только операция произведения. Поэтому если регулярное выражение α определяет конечно-автоматное множество X, то обращение множества X будет определяться регулярным выражением β, которое получается из α одновременной перестановкой всех сомножителей в произведениях. Иначе говоря, регулярное выражение β необходимо определить параллельно индуктивному определению выражения α. При этом пп. 1–3 в определении регулярного выражения α переносятся без изменения и на выражение β, а в п. 4 при переходе от α к β необходимо в произведении поменять порядок сомножителей.

Глава 3.

1.



2. Вес равен 4.

3. 1),3) да, 2),4),5) нет.

4. 1) 3 остаточные функции, которые отвечают входным словам $\Lambda, 0, 1$; 2) 2 остаточные функции, которые отвечают входным словам $\Lambda, 0$; 3) бесконечное число остаточных функций, каждая из которых определяется, например, словом 0^n, $(n \geqslant 0)$.

5. 1) вес равен 2, 2) вес равен 4, 3) вес равен 3, 4) вес равен 2.

6. 1)

$$\begin{cases} y_1(t) = \bar{q}_1(t-1), \\ q_1(t) = q_1(t-1) \to q_2(t-1), \\ q_2(t) = \bar{x}(t), \\ q_1(0) = q_2(0) = 0, \end{cases}$$

вес суперпозиции равен 3. 2) Вес равен 5.
3)

$$\begin{cases} y(t) = q_1(t-1), \\ q_1(t) = \bar{x}(t) \cdot q_2(t-1) \vee q_1(t-1), \\ q_2(t) = x(t), \\ q_1(0) = 0,\ q_2(0) = 1, \end{cases}$$

вес суперпозиции равен 2.

7. 1)

$$\begin{cases} y(t) = x_2(t), \\ q(t) = 1, \\ q(0) = 0, \end{cases}$$

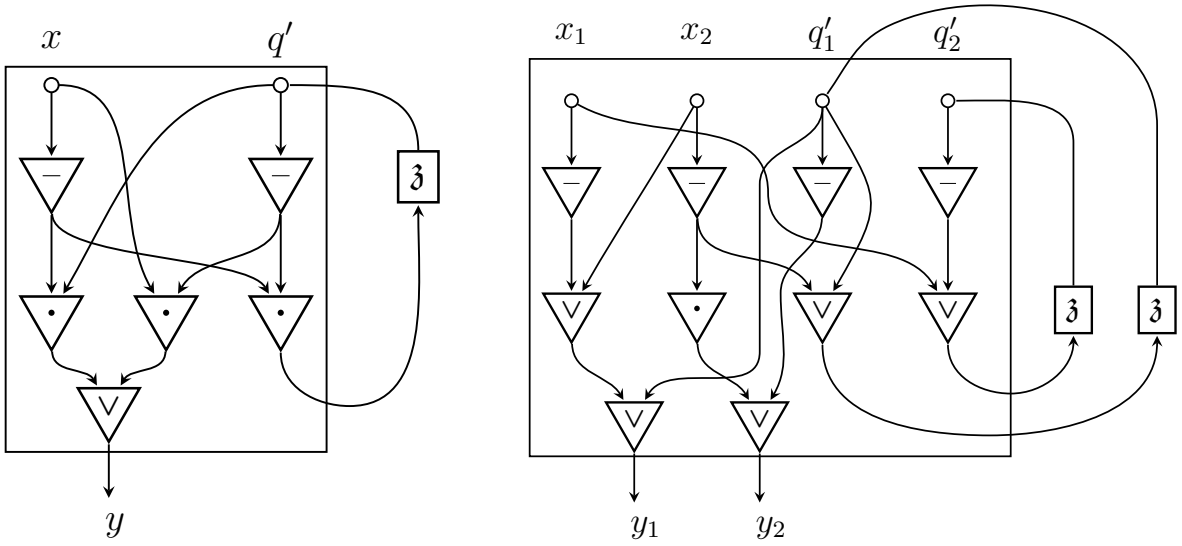
вес полученной функции равен 1.
2б)

$$\begin{cases} y(t) = 1, \\ q_1(t) = q_1(t-1) \oplus q_2(t-1), \\ q_2(t) = x_1(t) \oplus q_1(t-1), \\ q_1(0) = q_2(0) = 0, \end{cases}$$

вес полученной функции равен 1.

8. Подобные задачи традиционно решаются с помощью построения схем, состоящих из автоматных элементов. Так же, как для схем из функциональных элементов, строится ориентированный граф с входными полюсами (вершины, соответствующие входным переменным x), выходными полюсами (вершины, соответствующие выходным переменным y) и вершинами, соответствующими автоматным элементам $f_\vee, f_\&, f_-$, з (ниже в схеме мы оставляем только символы $\vee, \&$, $-$, з). В графе допускаются ориентированные циклы, которые обязательно должны проходить через элементы задержки. С использованием этих схем нетрудно восстановить последовательность применения операций суперпозиции и введения обратной связи, приводящую в итоге к построению искомых функций на основе заданных автоматных функций. В приводимых далее двух схемах прямоугольниками выделены части схем, которые не

содержат элементов задержки и которые определяют только истинностные функции.



9. В обоих случаях суперпозициями функции f_1 можно получить все истинностные функции из класса $P_{\kappa a,2}$. Далее подстановкой истинностных функций-констант в функцию f_2 образуем единичную задержку.

Глава 4.

1. Пусть программа машины Тьюринга $\mathcal{M}$ содержит команду (4.1), где $D = S$. Добавим к множеству состояний машины $\mathcal{M}$ новое состояние q'_s, а команду (4.1) заменим серией из $k + 2$ команд

$$a_i q_j \to a_l L q'_s, \quad a_m q'_s \to a_m R q_s \quad (l = 0, 1, \ldots, k).$$

Проделав это преобразование со всеми командами машины $\mathcal{M}$, содержащими символ S, получим программу машины Тьюринга $\mathcal{M}'$, которая эквивалентна машине $\mathcal{M}$.

2. Можно реализовать следующий алгоритм вычисления: стереть первую единицу основного кода, пробежать слева направо оставшийся массив из единиц (если он есть), заменить разделительный нуль единицей, вернуться к началу полученного массива из единиц, стереть в нем первую единицу, сдвинуться вправо на одну клетку и остановиться.

3. Может, только эти функции должны зависеть от различного числа переменных. Одна из простейших машин такого типа правильно вычисляет функции $x + 1$ и $x + y + 2$. Для этого она пробегает слева направо первый массив из единиц, заменяет стоящий справа 0 на 1 и затем возвращается на первую единицу вновь образовавшегося массива.

4. Можно. Достаточно для машины $\mathcal{M}'$ поменять ролями «лево» и «право». Иными словами, основной код набора $(x_1, \ldots, x_n)$ следует преобразовать в основной код набора $(x_n, \ldots, x_1)$, а в начальный момент времени головку машины $\mathcal{M}'$ установить на самую правую единицу основного кода.

6. Сначала следует записать символ 1 слева или справа от основного кода, отступив на одну пустую клетку. Затем (эти действия будут циклически повторяться) необходимо в основном коде числа x заменить две крайние единицы нулями (соблюдайте осторожность: они могут оказаться последними), добавить единицу к ранее поставленной единице и т.д.

7. Для случая $l = 3$ программа машины $\mathcal{M}_6$ выглядит так:

	q_1	q_2	q_3	q_4	q_5	q_6	q_6^1	q_6^2
0	$0Rq_2$	$0Rq_2$	$0Lq_7$	—	$0Lq_5$	$1Lq_6^1$	$1Lq_6^2$	$1Sq_1$
1	$1Rq_1$	$1Rq_3$	$1Lq_4$	$0Lq_5$	$1Lq_6$	$1Lq_6$	—	—

	q_7	q_8	q_9	q_9^1	q_9^2
0	—	$0Lq_8$	$1Lq_9^1$	$1Lq_9^2$	$1Sq_0$
1	$0Lq_8$	$1Lq_9$	$1Lq_9$	—	—

Двойными вертикальными отрезками отделены части машины $\mathcal{M}_6$, которые можно рассматривать как самостоятельные машины Тьюринга.

9. Для всякой машины Тьюринга $\mathcal{M}$ можно построить эквивалентную машину Тьюринга, если дописать к программе машины $\mathcal{M}$ произвольную «фиктивную» часть (состояния и команды, которые не связаны с состояниями машины $\mathcal{M}$).

10. Пусть функция $U(x, 2x)$ имеет всюду определенное вычислимое доопределение $V(x)$. Тогда функция $V(x)+1$ также всюду определена и вычислимаа. Возьмем такое n, чтобы вычислимая функция $U(n, 2x)$ как функция от x совпадала с функцией $V(x) + 1$, т.е. $U(n, 2x) = V(x) + 1$. Получим противоречие, если положим $x = n$.

11. Пусть функция $U(n, x)$ вычислима на машине Тьюринга $\mathcal{U}$, а функция $U(n_0, x)$ определена хотя бы в одной точке. Рассмотрим алгоритм вычисления всюду определенной функции $f(n, x, t)$, которая перечисляет множество M. Запускаем машину $\mathcal{U}$ на паре (n, x) и «прослеживаем» вычисление в течение t тактов. Если машина $\mathcal{U}$ за это время заканчивает вычисление, то полагаем $f(n, x, t) = n$. В противном случае полагаем $f(n, x, t) = n_0$.

12. 1. Пусть $p(x) = a_k x^k + \ldots + a_1 x + a_0$, где $k > 1$, $a_k \neq 0$, и $a = \max\{a_k, \ldots, a_1, a_0\}$. Тогда возможное решение уравнения $y = p(x)$

заключено в пределах

$$\sqrt[k]{\frac{y}{a(k+1)}} \leqslant x \leqslant \sqrt[k]{\frac{y}{a_k}}.$$

Следовательно, длина $|x|$ двоичной записи числа x будет находиться в пределах

$$[(|y| - |a(k+1)| - 1)/k] \leqslant |x| \leqslant \lceil (|y| - |a_k| + 1)/k) \rceil.$$

Из этой формулы видно, что интервал для возможных значений величины $|x|$ имеет константную длину (зависящую только от полинома p). Для соответствующих значений переменной x необходимо вычислить значения полинома p и сравнить их со значением y. Это приводит к полиномиальному переборному алгоритму.
2. Решение этой задачи сводится к решению задачи из п.1.
3. Для отыскания решения уравнения $y = c^x$ следут, например, перебрать все значения x, не превосходящие $[\log_2 y / \log_2 c] \leqslant |y|+1$, и проверить выполнение равенства $y = c^x$. Поскольку каждое умножение на константу c выполняется за не более чем квадратичное (от длины записи аргумента) время, приходим к полиномиальному времени решения данной задачи.
4. Необходимо предварительно привести квадратную матрицу к диагональному виду.

13. 1. Необходимо сначала выделить существенные переменные функции, а затем воспользоваться тем фактом, что линейная функция равна 1 на наборах (для существенных переменных) либо только с нечетным, либо только с четным числом единиц.
2. В случае задания таблицей значений можно впрямую воспользоваться определением монотонной функции. В случае полинома Жегалкина необходимо применить следующее утверждение: функция $F(x_1, \ldots, x_n)$ монотонна тогда и только тогда, когда при любом i $(1 \leqslant i \leqslant n)$ справедливо тождество

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1} \ldots, x_n) \cdot (f(x_1, \ldots, x_{i-1}, 1, x_{i+1} \ldots, x_n) \oplus 1) = 0.$$

3. Можно проверить значения функции на всех парах противоположных наборов.

14. 2. Пусть $f(x_1, \ldots, x_n)$ — данная функция. Для проверки нелинейности функции f достаточно найти такие числа i, j $(1 \leqslant i < j \leqslant n)$ и такой набор значений переменных функции f, отличных от x_i, x_j, что

при подстановке этих значений в ДНФ функции f образуется нелинейная функция от переменных x_i, x_j.

15. 1–3. В каждой из задач недетерминированно порождается требуемое множество или функция, а затем детерминированно и за полиномиальное время проверяется соответствие найденного объекта имеющимся условиям.

16. 1. Покажем, что задача ВЫП P-сводится к задаче КЛИКА. Пусть $K = D_1 \,\&\, D_2 \,\&\, \ldots \& D_s$ — КНФ с дизъюнктами $D_i = t_{i1} \vee t_{i2} \vee \ldots \vee t_{im_i}$. Образуем граф $G = (V, E)$. Каждому литералу t_{ij} из K сопоставим вершину v_{ij} множества V. Далее полагаем

$$(v_{i_1 j_1}, v_{i_2 j_2}) \in E \Leftrightarrow i_1 \neq j_1 \text{ и } t_{i_2 j_2} \neq \bar{t}_{i_1 j_1}.$$

Тогда КНФ K будет выполнимой в том и только том случае, когда в графе G имеется клика с s вершинами. При этом очевидно, что граф G эффективно строится по КНФ K за полиномиальное время.

2. Установим P-сводимость задачи КЛИКА к задаче ВЕРШИННОЕ ПОКРЫТИЕ. Пусть $G = (V, E)$ — граф, $n = |V|$ и k — натуральное число. Рассмотрим граф $\bar{G} = (V, E')$ — дополнение графа G, где при $u \neq v$ имеем $(u, v) \in E' \Leftrightarrow (u, v) \notin E$. Тогда граф G содержит клику с k вершинами в том и только том случае, когда граф $\bar{G}$ имеет вершинное покрытие с $n - k$ вершинами.

17. 1. Сначала заметим, что NP-полной является задача о существовании одного набора, на котором ДНФ обращается в нуль. В самом деле, если K — КНФ, то $\bar{K}$ легко приводится к ДНФ D того же размера. При этом K выполнима в том и только том случае, когда ДНФ D обращается в нуль на некотором наборе. Переходя к двум противоположным наборам, предположим, что задана ДНФ D от переменных $x_1, \ldots, x_n$ и переменные $t, y_1, \ldots, y_n$ отличны от переменных $x_1, \ldots, x_n$. Пусть ДНФ D_1 получена из ДНФ D заменой каждого литерала $\bar{x}_i$ переменной y_i и $D' = t \,\&\, D_1 \vee x_1 \,\&\, y_1 \vee \ldots \vee x_n \,\&\, y_n$. Тогда ДНФ D обращается в нуль в том только том случае, когда ДНФ D' обращается в нуль на паре противоположных наборов.

2. Пусть D — ДНФ функции, не равной тождественно нулю, t - переменная, которая не встречается в D. Тогда ДНФ D обращается в нуль на некотором наборе в том и только том случае, когда ДНФ функции $D \,\&\, t$ реализует нелинейную функцию.

3. Пусть D и t такие же, как в предыдущей задаче. Тогда ДНФ D обращается в нуль на некотором наборе в том и только том случае, когда

ДНФ $D \vee \bar{t}$ реализует немонотонную функцию.
4. Используется конструкция из задачи 2.

18. Функцию $I_2^2(x_1, x_2)$ можно, например, получить примитивной рекурсией и суперпозицией из функций $0, x+1$ и I_3^3.

20. Можно считать, что $d > 1$. Функция $\mathrm{rm}(x, d)$ периодическая с периодом d. Любая периодическая функция $f(x)$ с периодом d может быть получена из функции $\mathrm{rm}(x, d)$ подходящей заменой ее значений $0, 1, \ldots, d-1$ (с помощью, например, утверждения (4.2)).

21. Можно воспользоваться нумерационной функцией $c(x, y)$ (см. § 9) и образовать вспомогательную функцию

$$f'(x_1, \ldots, x_n) = c^{d+1}(f(x_1, \ldots, x_n), \ldots, f(x_1, \ldots, x_{n-1}, x_n + d)).$$

22. Следует n раз применить операцию ораниченного суммирования к функции $\overline{\mathrm{sg}}\,|P(x_1, \ldots, x_n) - Q(x_1, \ldots, x_n)|$.

23. Пусть $h(y, z)$ — характеристическая функция отношения $g(z) = y$. Тогда

$$f(x, y) = \sum_{z \leqslant x} (z \cdot \overline{\mathrm{sg}} \sum_{i<z} h(y, i))$$

(операция $\sum_{i<z}$ определяется так же, как и операция $\sum_{i \leqslant z}$).

24. Легко показать, что для всякого x число $p(x+1)$ не превосходит величины $p(x)^{p(x)}$. Поэтому функцию $p(x)$ можно определить следующей примитивной рекурсией:

$$\begin{cases} p(0) = 2, \\ p(x+1) = \text{наименьшему } z, \text{ такому, что } p(x) < z \leqslant p(x)^{p(x)} \\ \qquad\qquad \text{и } \mathrm{Pr}(z) \text{ истинно.} \end{cases}$$

25. Очевидно, что $f(0) = 4$. Далее, $f(x+1)$ равно такому (единственному) числу a $(0 \leqslant a \leqslant 9)$, что

$$\left(1 + \frac{f(0)}{10} + \ldots + \frac{f(x)}{10^{x+1}} + \frac{a}{10^{x+2}}\right)^2 < 2$$

и

$$\left(1 + \frac{f(0)}{10} + \ldots + \frac{f(x)}{10^{x+1}} + \frac{a+1}{10^{x+2}}\right)^2 > 2.$$

Эти два соотношения можно переписать в виде

$$(10^{x+2} + f(0) \cdot 10^{x+1} + \ldots + f(x) \cdot 10 + a)^2 < 2 \cdot 10^{2(x+2)},$$

$$(10^{x+2} + f(0) \cdot 10^{x+1} + \ldots + f(x) \cdot 10 + a + 1)^2 > 2 \cdot 10^{2(x+2)}.$$

Отсюда нетрудно уже получить схему примитивной рекурсии, определяющую функцию $f(x)$.

26. Если $g(0,\ldots,0)=a$, то для функции f, полученной согласно (4.6), имеем $f(0,\ldots,0,a)=0$.

27.

$$(\mu z)(x+z=y)=\begin{cases} y-x, & \text{если } y\geqslant x,\\ \text{не определено}, & \text{если } y<x,\end{cases}$$

$$(\mu z)(x\cdot z=y)=\begin{cases} 0, & \text{если } y=0,\\ y/x, & \text{если } y\neq 0 \text{ и } y \text{ делится нацело на } x,\\ \text{не определено} & \text{в остальных случаях},\end{cases}$$

$$(\mu z)(x-z=y)=\begin{cases} x-y, & \text{если } x\geqslant y,\\ \text{не определено}, & \text{если } x<y,\end{cases}$$

$$(\mu z)(z-x=y)=\begin{cases} y, & \text{если } x=0,\\ \text{не определено}, & \text{если } x\neq 0,\end{cases}$$

$(\mu z)(x/z=y)$ есть нигде не определенная функция,

$$(\mu z)(z/x=y)=\begin{cases} xy, & \text{если } x\neq 0,\\ \text{не определено}, & \text{если } x=0.\end{cases}$$

28. $g_1(x,y)=|x-y|$, $g_2(x,y)=[x/y]$ при условии, что $[x/0]=0$.

29. Определим примитивно рекурсивную функция $g(x)$:

$$g(x)=a_1\cdot\overline{\operatorname{sg}}\,|x-1|+\ldots+a_s\cdot\overline{\operatorname{sg}}\,|x-s|+a_1\cdot\operatorname{sg}|x-1|\cdot\ldots\cdot\operatorname{sg}|x-s|.$$

Она принимает лишь значения $a_1,\ldots,a_s$. Имеем

$$f(x)=\operatorname{sg}((\mu y)(g(y)=x)+1).$$

30. Имеем $g(x)=\overline{\operatorname{sg}}((\mu y)(l_1(x,y)=1)+1)$.

31. Расположим все пары чисел из N в следующем порядке:

$$(0,0);\ (0,1),(1,0);\ (0,2),(1,1),(2,0);\ (0,3),\ldots$$

Соответствующая этому порядку нумерующая функция есть

$$x+\frac{(x+y)(x+y+1)}{2}=\frac{(x+y)^2+3x+y}{2}.$$

«Обратные» функции $l(v)$ и $r(v)$ определяются формулами

$$l(v)=v\dot{-}\frac{1}{2}\left[\frac{[\sqrt{8v+1}]+1}{2}\right]\left[\frac{[\sqrt{8v+1}]\dot{-}1}{2}\right],$$

$$r(v) = \left[\frac{[\sqrt{8v+1}]+1}{2}\right] \dot{-} (l(v)+1).$$

32. Пусть $f(x_1, \ldots, x_n)$ — примитивно рекурсивная функция, которая вычислима на машине Тьюринга за полиномиальное время (при двоичном кодировании аргументов). Тогда она, конечно, будет вычислима на подходящей машине Тьюринга $\mathcal{M}$ за полиномиальное время $p(x_1, \ldots, x_n)$ при кодировании аргументов основным кодом. Следовательно, примитивно рекурсивная функция $\mathrm{Code}(x_1, \ldots, x_n, p(x_1, \ldots, x_n))$ даст код заключительной конфигурации машины Тьюринга $\mathcal{M}$. Далее, как в доказательстве теоремы 4.6, примитивно рекурсивным образом «извлекаем» из этого кода значение функции f. Такая же идея применима при рассмотрении отношений из класса NP.

33. Пусть частично рекурсивная функция $f(x_1, \ldots, x_n)$ перечисляет непустое множество чисел и a — некоторый элемент этого множества. Тогда (см. представление Клини в теореме 4.6) это множество перечислимо примитивно рекурсивной функцией $f'(x_1, \ldots, x_n, y)$, определяемой следующими равенствами:

$$f'(x_1, \ldots, x_n, y) = \begin{cases} G(x_1, \ldots, x_n, y), & \text{если } H_f(x_1, \ldots, x_n, y) = 0, \\ a & \text{в противном случае.} \end{cases}$$

Список литературы

[1] Алексеев В.Б. Лекции по дискретной математике. М.: ИНФРА-М, 2012. 90 с.

[2] Васильев Ю.Л., Ветухновский Ф.Я., Глаголев В.В., Журавлев Ю.И., Левенштейн В.И., Яблонский С.В. Дискретная математика и математические вопросы кибернетики, I. М.: Наука, 1974. С. 9–98.

[3] Гаврилов Г.П., Сапоженко А.А. Задачи и упражнения по дискретной математике. М.: Физматлит, 2004. 416 с.

[4] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.

[5] Кук С.А. Сложность процедур вывода теорем // Кибернетический сборник. Новая серия. Вып. 12. М.: Мир, 1975. С. 5–15.

[6] Марченков С.С. Основы теории булевых функций. М.: Физматлит, 2014. 136 с.

[7] Яблонский С.В. Функциональные построения в k-значной логике // Труды Матем. ин-та им. В.А. Стеклова АН СССР. 1958. Т. LI. С. 5–142.

[8] Яблонский С.В. Введение в дискретную математику. М.: Высшая школа, 2003. 384 с.

[9] Янов Ю.И., Мучник А.А. О существовании k-значных замкнутых классов, не имеющих базиса // ДАН СССР. 1959. Т. 127, № 1. С. 44–46.

[10] Post E.L. Introduction to a general theory of elementary propositions // Amer. J. Math. 1921. V. 43, № 4. P. 163–185.

[11] Post E.L. Two-valued iterative systems of mathematical logic // Annals of Math. Studies. Princeton Univ. Press, 1941. V. 5. P. 1–122.