Федеральное агентство по образованию

Московский инженерно-физический институт
(государственный университет)

М.А. Короткова

МАТЕМАТИЧЕСКАЯ ТЕОРИЯ АВТОМАТОВ

Учебное пособие

Рекомендовано УМО «Ядерные физика и технологии»
в качестве учебного пособия для студентов
высших учебных заведений

Москва 2008

УДК 519.713+ 519.725
ББК 22.18
К 687

Короткова М.А. Математическая теория автоматов. *Учебное пособие.* М.: МИФИ, 2008. – 116 с.

Рассматриваются вопросы представления детерминированных функций k-значных логик, способы задания, анализа и синтеза автоматов. Рассмотрены также базовые понятия теории кодирования, включая вопросы распознавания кодов автоматами. Пособие предназначено для студентов, обучаемых по специальности «Прикладная математика и информатика» и изучающих курс «Математическая теория автоматов». Предлагаемое пособие будет полезно также студентам третьего курса факультета Кибернетики, изучающим математическую лингвистику и теорию автоматов. Пособие может быть рекомендовано всем интересующимся теорией автоматов.

Пособие подготовлено в рамках Инновационной образовательной программы

Рецензент канд. техн. наук, доц. Г.М. Сергиевский

ISBN 978-5-7262-1005-6

Редактор Шумакова Н.В.
Оригинал-макет изготовлен Коротковой М.А.

Подписано в печать 20.11.2008 Формат 60×84 1/16
Печ.л. 6,25 Уч.-изд.л. 6,25 Тираж 150 экз.
Изд. № 4/32 Заказ №

Московский инженерно-физический институт
(государственный университет).
115409, Москва, Каширское ш., 31

Типография издательства «Тровант».
г. Троицк Московской обл.

Содержание

Предисловие

Существуют различные традиции в изложении теории автоматов. В данном пособии сделана попытка совмещения описания математического базиса разработки автоматных моделей – ***k***- значных логик и таких технических аспектов, как построение функций выхода и перехода при заданном кодировании состояний и сигналов. Рассматриваются также общие вопросы кодирования сигналов в автоматах, что позволяет пояснить вопросы синтеза автоматов и объяснить принцип построения декодирующего автомата для префиксного кодирования, выявляя взаимосвязь различных аспектов рассматриваемой теории. Изложение теории сопровождается многочисленными иллюстрациями и примерами.

Настоящее пособие предполагает, что читатель знаком с основными понятиями теории множеств и математической логики, в частности, алгебры высказываний.

Для лучшего понимания излагаемого материала в каждой главе приведены вопросы и упражнения. Значительная доля их требует только внимательного изучения текста главы или применения описанных алгоритмов, тем не менее ответы на некоторые вопросы требуют сопоставления изложенного в разных главах.

В конце пособия приводится список использованной и рекомендуемой литературы.

ГЛАВА 1. ДЕТЕРМИНИРОВАННЫЕ ФУНКЦИИ И СПОСОБЫ ИХ ЗАДАНИЯ

В данной главе рассматриваются логики, при условии, что $k \geq 3$. Предполагается, что читатели знакомы с функциями обычной булевой логики (k=2) и их свойствами [1,2].

k- значные логики вводятся как обобщение двузначной логики.

С одной стороны, для ***k***-значных логик сохраняются некоторые важные свойства двузначных логик, также сохраняются многие важные результаты, полученные для двузначных логик.

С другой стороны, в ***k***-значных логиках наблюдаются явления, отличающие их от двузначной логики. Последнее обстоятельство обусловливает необходимость рассмотрения этих логик в данном пособии. Рассматриваются не все особенности ***k***-значных логик, а только те базовые свойства, знание которых необходимо для изучения остальных разделов курса.

Функции *k*-значной логики. Формулы и реализация функций формулами

В формулах используется алфавит $U=\{u_1,u_2,\ldots,u_m,\ldots\}$ – исходный алфавит переменных (аргументов).

Будем рассматривать функции $f\left(u_{i_1},u_{i_2},...,u_{i_n}\right)$ ($u_{i_j} \neq u_{i_l}$ при $j \neq l$), аргументы которых определены на $E_k=\{0,1,\ldots,k\text{-}1\}$, такие, что $f\left(\alpha_1,\alpha_2,...,\alpha_n\right) \in E_k$ при $\alpha_i \in E_k, i \in [1,n]$.

Например, для трехзначной логики $E_3=\{0, 1, 2\}$. Аргументы и значения функций – из этого множества.

Для упрощения записи функций для обозначения аргументов используются символы ***x,y,z*** с индексами или без них.

Считаем, что функция $f\left(x_1,x_2,...,x_n\right)$ полностью определена, если задана её таблица.

Например, зададим таблицей функцию трёхзначной логики $\varphi_1(x_1,x_2)$ от двух аргументов (табл.1):

Таблица 1

x_1 x_2	$\varphi_1(x_1,x_2)$
0 0	1
0 1	0
0 2	2
1 0	1
1 1	1
1 2	0
2 0	2
2 1	2
2 2	0

Функция от одной переменной может быть записана как подстановка:

$$S(x) = \begin{pmatrix} 0\,1 \ldots n \\ i_0\, i_1 \ldots i_n \end{pmatrix}.$$

Например, можно так описать функцию одного аргумента для 5-тизначной логики: $\varphi_2(x) = \begin{pmatrix} 0\,1\,2\,3\,4 \\ 1\,2\,3\,4\,0 \end{pmatrix}$.

Наборы аргументов для функции от **n** аргументов можно рассматривать как разложения в k-ричной системе чисел 0, 1,…, k^n-1. Т.е. набору аргументов (2, 1) соответствует число 21.

Обозначим $\mathbf{P}_k$ множество всех функций k-значной логики (множество функций включает в себя также набор констант, которые можно рассматривать, например, как функции без аргументов).

Теорема 1. Число всех функций $\mathbf{P}_k$, зависящих от **n** переменных, равно k^{k^n}.

Доказательство. В самом деле, каждая функция однозначно определяется таблицей истинности. Строчек в таблице k^n. Каждому набору переменных сопоставляется одно значение функции, кото-

рое принадлежит множеству E_k. Функции различны, если они различаются хотя бы для одного набора аргументов. Различных значений функции – ***k***. Известно, что всего функций из B в A равно $|A|^{|B|}$, поэтому всех функций ***k***-значной логики k^{k^n}.

Это число довольно велико, например, трёхзначных функций от двух аргументов – 3^9=19683.

Функции часто задают с помощью алгоритма их вычисления, например, $\max\{x_1, x_2, ..., x_n\}$.

Говорят, что функция $f(x_1, x_2, ..., x_n)$ существенным образом зависит от переменной x_i, если существуют значения **a** и **b**, принадлежащие множеству {0,…, ***k***-1} такие, что
$f(x_1, ..., x_{i-1}, a, x_{i+1}, ..., x_n) \neq f(x_1, ..., x_{i-1}, b, x_{i+1}, ..., x_n)$.

Рассмотрим примеры некоторых конкретных функций из $\boldsymbol{P_k}$, которые можно считать «элементарными» функциями.

1. $\overline{x} = x + 1 (\mathrm{mod}\, k)$. Функция является обобщением отрицания в смысле «циклического» сдвига значений.

2. $Nx = k - 1 - x$, отрицание Лукасевича, часто обозначается также ~*x*. Является другим обобщением отрицания в смысле «зеркального» отражения значений. Следует обратить внимание на то, что в двузначной логике оба приведённых обобщения отрицания дают обычное отрицание булевой алгебры.

3. $I_i(x) = \begin{cases} k-1 \text{ при } x = i, \\ 0 \text{ при } x \neq i. \end{cases} (i = 0, 1, ..., k-1)$

Эта функция при $i \neq$ ***k***-1 тоже обобщает некоторые свойства отрицания.

4. $j_i(x) = \begin{cases} 1 \text{ при } x = i, \\ 0 \text{ при } x \neq i. \end{cases}$

Характеристическая функция значения ***i***, при ***i*≠*k***-1 представляет собой обобщение отрицания.

5. min(***x***$_1$,***x***$_2$) – обобщение конъюнкции.

6. ***x***$_1$***x***$_2$(***mod k***) – также обобщение конъюнкции.

7. $\max(\boldsymbol{x}_1, \boldsymbol{x}_2)$ – обобщение дизъюнкции.

8. $\boldsymbol{x}_1+\boldsymbol{x}_2(\boldsymbol{mod\ k})$ – обобщение жегалкинского сложения.

Функции, представленные в пунктах 5-8 в случае бинарной логики ведут себя так же, как соответствующие функции булевой алгебры.

Примеры этих функций для $\boldsymbol{k}$=3 приводятся в таблицах 2 (унарные функции) и 3 (функции двух аргументов).

Таблица 2

$\boldsymbol{x}$	$\bar{x}$	$\sim x$	$\boldsymbol{I}_1(\boldsymbol{x})$	$\boldsymbol{I}_0(\boldsymbol{x})$	$j_1(x)$	$j_0(x)$
0	1	2	0	2	0	1
1	2	1	2	0	1	0
2	0	0	0	0	0	0

Таблица 3

$\boldsymbol{x}_1$ $\boldsymbol{x}_2$	$\min(\boldsymbol{x}_1,\boldsymbol{x}_2)$	$\boldsymbol{x}_1+\boldsymbol{x}_2(\boldsymbol{mod\ k})$	$\max(\boldsymbol{x}_1,\boldsymbol{x}_2)$	$\boldsymbol{x}_1\boldsymbol{x}_2(\boldsymbol{mod\ k})$
0 0	0	0	0	0
0 1	0	1	1	0
0 2	0	2	2	0
1 0	0	1	1	0
1 1	1	2	1	1
1 2	1	0	2	2
2 0	0	2	2	0
2 1	1	0	2	2
2 2	2	1	2	1

Определим формулу над множеством функций $\boldsymbol{S}$.

Формулой над множеством функций $S = \{\varphi_1(x_1,...,x_{k_1}), \varphi_2(x_1,...,x_{k_2}),..., \varphi_l(x_1,...,x_{k_l})\}$ называется любая функция $\boldsymbol{f}$, получаемая

1) из $\varphi_j(x_1,...,x_{k_j}) \in \boldsymbol{S}, \boldsymbol{j} \in \{1,\boldsymbol{l}\}$ переименованием переменных;

2) подстановкой вместо некоторых переменных функции $\varphi_\alpha(x_1,...,x_{k_\alpha})$, $\alpha\in\{1,l\}$ функций $\varphi_j\in S$;

3) многократным применением суперпозиции.

Каждой формуле однозначно сопоставляется функция $\boldsymbol{f}$ из $\boldsymbol{P_k}$. Говорят, что формула реализует функцию. Формулы считаются эквивалентными, если реализуемые ими функции равны.

Например, эквивалентны функции min ($\boldsymbol{x}_1$, min($\boldsymbol{x}_1$,$\boldsymbol{x}_2$)) и min($\boldsymbol{x}_1$,$\boldsymbol{x}_2$).

Замыканием [$\boldsymbol{K}$] множества функций $\boldsymbol{K}$ алгебры логики называется множество всех функций алгебры логики, являющихся суперпозициями множества функций из $\boldsymbol{K}$.

Множество функций $\boldsymbol{K}$ называется (функционально) замкнутым, если [$\boldsymbol{K}$]=$\boldsymbol{K}$. Замкнутые множества функций также называют замкнутыми классами.

Подмножество $\boldsymbol{P}$ замкнутого множества $\boldsymbol{K}$ называется функционально полным в $\boldsymbol{K}$, если [$\boldsymbol{P}$]=$\boldsymbol{K}$. Функционально полное множество называется базисом, если никакое его подмножество функционально полным не является (т.е. базис – минимальное по включению полное множество функций).

Класс $\boldsymbol{P}_2$ – класс всех булевых функций. Для этого класса существует достаточно много базисов[1,2], например, $\{\bar{}, \&\}$ или $\{\circ\}$(функция Вебба, $x_1 \circ x_2 = \overline{x_1} \,\&\, \overline{x_2}$).

Основные свойства элементарных функций k-значной логики

Здесь $\boldsymbol{x}_1\circ\boldsymbol{x}_2$ обозначает любую из функций $\boldsymbol{x}_1+\boldsymbol{x}_2$(***mod k***), $\boldsymbol{x}_1\,\boldsymbol{x}_2$ (***mod k***), min($\boldsymbol{x}_1$, $\boldsymbol{x}_2$), max($\boldsymbol{x}_1$,$\boldsymbol{x}_2$),

1. Ассоциативность: $(\boldsymbol{x}_1\circ\boldsymbol{x}_2)\circ\boldsymbol{x}_3=\boldsymbol{x}_1\circ(\boldsymbol{x}_2\circ\boldsymbol{x}_3)$.
2. Коммутативность: $\boldsymbol{x}_1\circ\boldsymbol{x}_2=\boldsymbol{x}_2\circ\boldsymbol{x}_1$.
3. Для закона двойного дополнения существует аналог: $\sim(\sim x)=x$, но $\overline{\overline{x}} \neq x$ при $\boldsymbol{k}\geq 3$.
4. Для законов Де-Моргана также существуют аналоги:

$\sim \min(x_1, x_2) = \max(\sim x_1, \sim x_2)$, $\sim \max(x_1, x_2) = \min(\sim x_1, \sim x_2)$, но $\overline{\min(x_1, x_2)} \neq \max(\overline{x_1}, \overline{x_2})$ при $k \geq 3$.

Правила преобразования формул для системы S_k

Рассматриваются правила для системы $S_k = \{0, 1, \ldots, k-1, I_0(x), I_1(x), \ldots, I_{k-1}(x), \min(x_1, x_2), \max(x_1, x_2)\}$ (далее $\min(x_1, x_2)$ обозначается как $x_1 \& x_2$ или просто $x_1 x_2$, $\max(x_1, x_2)$ – как $x_1 \vee x_2$).

1. Правила спуска I «вглубь» формулы:

$$I_\sigma(I_\delta(x)) = \begin{cases} I_0(x) \vee \ldots \vee I_{\delta-1}(x) \vee I_{\delta+1}(x) \vee \ldots \vee I_{k-1}(x), \text{ при } \sigma = 0, \\ 0, \text{ при } 0 < \sigma < k-1, \\ I_\delta(x), \text{ при } \sigma = k-1. \end{cases}$$

$I_\sigma(x_1 \& x_2) = I_\sigma(x_1) \& (I_\sigma(x_2) \vee \ldots \vee I_{k-1}(x_2)) \vee I_\sigma(x_2)(I_\sigma(x_1) \vee \ldots \vee I_{k-1}(x_1))$

$I_\sigma(x_1 \vee x_2) = I_\sigma(x_1) \& (I_0(x_2) \vee \ldots \vee I_\sigma(x_2)) \vee I_\sigma(x_2)(I_0(x_1) \vee \ldots \vee I_\sigma(x_1))$.

2. Законы дистрибутивности:

$(x_1 \vee x_2) \& x_3 = x_1 \& x_3 \vee x_2 \& x_3$,

$x_1 \vee x_2 \& x_3 = (x_1 \vee x_2) \& (x_1 \vee x_3)$.

3. Исключение чистых вхождений переменных:

$x = 1 \& I_1(x) \vee 2 \& I_2(x) \vee \ldots \vee I_{k-1}(x)$.

4. Правило введения переменных:

$x_1 = x_1(I_0(x_2) \vee I_1(x_2) \vee \ldots \vee I_{k-1}(x_2))$.

5. Правила упрощений: $I_\sigma(x)\, I_\tau(x) = \begin{cases} I_\sigma(x), \text{ при } \sigma = \tau, \\ 0, \text{ при } \sigma \neq \tau. \end{cases}$

$(k-1)\, x = x$, $\quad 0\, x = 0$, $\quad (k-1) \vee x = k-1$, $\quad 0 \vee x = x$.

Для функций k-значной логики существует некоторый аналог ДНФ:

$$f(x_1, x_2, \ldots, x_n) = \bigvee_{(\sigma_1, \sigma_2, \ldots, \sigma_n)} I_{\sigma_1}(x_1) \& I_{\sigma_2}(x_1) \& \ldots \& I_{\sigma_n}(x_1) \& f(\sigma_1, \sigma_2, \ldots, \sigma_n)$$

Дизъюнкция проводится по всем возможным наборам $\sigma_1, \sigma_2, \ldots, \sigma_n$.

Очевидно, что каждая формула может быть преобразована к такому виду, что следует из однозначности построения аналога ДНФ по таблице функции.

Полнота системы функций

Система S функций $f_1, \dots, f_s$ называется (функционально) полной в классе P_k, если любая функция из P_k может быть выражена как суперпозиция функций системы S.

1. $S = P_k$ является полной по определению.
2. **Теорема 2**. Система функций S_k является полной в классе функций P_k. Доказательство можно провести через аналог ДНФ.
3. Система $S'_k = \{\overline{x}, \max(x_1, x_2)\}$ является полной в P_k.

Очертим путь доказательства полноты. Так, $\overline{x} = x + 1, \overline{\overline{x}} = x + 2,$ и т.д. Затем можно получить первую константу $\max\{x, x+1, \dots x + k-1\} = k-1$, после чего очевидным способом получают остальные константы.

$I_i(x) = 1 + \max\limits_{\alpha \neq k-1-i} \{x + \alpha\}$, затем получаются функция минимума.

Подробности можно посмотреть в [1].

4. Существует полная система из одной функции, это функция Вебба $V_k = \max(x_1, x_2) + 1$. Она также является полной, доказательство её полноты доказывается через полноту системы 3.

Ограниченно-детерминированные (автоматные) функции с операциями

Считаем известными две функциональные системы с операциями:

(P_2, C) – булева алгебра логики, система функций алгебры логики с операцией суперпозиции;

(P_k, C) – k-значная логика, т.е. система функций k-значной логики с операцией суперпозиции.

Знание этих систем позволяет перейти к рассмотрению более сложных систем.

$(P_{\aleph_0}, C)$ – счётно-значная логика, т.е. система, содержащая константы 0, 1, …, $\boldsymbol{k}$, … и функции f($\boldsymbol{x}_1$, …, $\boldsymbol{x}_n$), переменные которых определены на расширенном натуральном ряде $E_{\aleph_0}$ ={0, 1, 2, …} , а сами функции принимают значения из $E_{\aleph_0}$, с операцией суперпозиции. Например, этому классу принадлежат все примитивно-рекурсивные функции.

$(P_{\aleph}, C)$ – континуумзначная логика, т.е. система, содержащая константы из [0,1] и функции, переменные которых определены на интервале [0,1], и сами принимают значения из [0,1], с операцией суперпозиции. Здесь $\aleph = 2^{\aleph_0}$.

Детерминированные функции

Рассмотрим следующую разновидность континуумзначной логики. Вместо x∈[0,1] будем рассматривать множество $E_{\aleph,k}$ всех $\boldsymbol{k}$-значных последовательностей α вида α ={α(1), α(2),…α($\boldsymbol{m}$),…}, где α($\boldsymbol{m}$)∈$\boldsymbol{E_k}$ для всех $\boldsymbol{m}$ ($\boldsymbol{m}$=1,2,…). Тогда $P_{\aleph,k}$ состоит из всех констант из $E_{\aleph,k}$ (т.е. наборов из $E_{\aleph,k}$) и множества всех функций $\boldsymbol{f}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, определённых на наборах ($\alpha_1, \alpha_2, \ldots \alpha_n$), $\alpha_i \in E_{\aleph,k}$ и принимающих значения из $E_{\aleph,k}$. Константы могут рассматриваться как функции без аргументов.

Например, при $\boldsymbol{k}$ =2 можно определить функцию $f_1(\boldsymbol{x})$:

$$f_1(\alpha) = \begin{cases} \{0,0,0,...\}, \text{ если } \alpha = \{0,0,0,...\}, \\ \{1,1,1,...\}, \text{ если } \alpha \neq \{0,0,0,...\}. \end{cases}$$

Построенная таким образом функция $f_1(\boldsymbol{x}) \in E_{\aleph,2}$.

В силу бесконечности последовательности аргументов функции из $E_{\aleph,k}$ функции не могут быть заданы таблицей. Далее для представления функций используется векторная запись, определённая ниже.

Набор из $\boldsymbol{n}$ переменных $(\boldsymbol{x}_1,\ldots, x_n)$ будем обозначать как $\overline{X}$, тогда $\boldsymbol{f}(\boldsymbol{x}_1,\ldots, \boldsymbol{x}_n)$ будет обозначаться как $\boldsymbol{f}(\overline{X})$.

Значение $\overline{X}$ – вектор (набор) $\overline{\alpha}=(\alpha_1, \alpha_2,.. \alpha_{\boldsymbol{n}})$, компоненты которого – последовательности значности $\boldsymbol{k}$: $\alpha_{\boldsymbol{i}} =\{\alpha_{\boldsymbol{i}}(1), \alpha_{\boldsymbol{i}}(2), \ldots \alpha_{\boldsymbol{i}}(m), \ldots\}$, $\boldsymbol{i}=1,2,\ldots\boldsymbol{n}$.

Тогда трактуем $\overline{\alpha}$ как последовательности векторов $\overline{\alpha} =$ $=\{\overline{\alpha}\ (1), \overline{\alpha}\ (2),\ldots\ \overline{\alpha}\ (\boldsymbol{m}),\ldots\}$, где $\overline{\alpha}\,(\boldsymbol{m})= (\alpha_1(\boldsymbol{m}), \alpha_2(\boldsymbol{m}) \ldots \alpha_{\boldsymbol{n}}(m))$, $\boldsymbol{m} = 1,2,\ldots$

Т.е. считаем, что выполнено тождество:

$(\{\alpha_1(1), \alpha_1(2),\ldots \alpha_1(\boldsymbol{m}),\ldots\}$,$\{\alpha_2(1), \alpha_2(2),\ldots \alpha_2(\boldsymbol{m}),\ldots\}$, … ,$\{\alpha_n(1), \alpha_n(2),\ldots \alpha_{\boldsymbol{n}}(\boldsymbol{m}),\ldots\})=\{(\alpha_1(1), \alpha_2(1),\ldots, \alpha_{\boldsymbol{n}}(1)), (\alpha_1(2), \alpha_2(2),\ldots, \alpha_{\boldsymbol{n}}(2)), \ldots, (\alpha_1(\boldsymbol{m}), \alpha_2(\boldsymbol{m}), \ldots, \alpha_{\boldsymbol{n}}(\boldsymbol{m})),\ldots\}$.

Полученную последовательность можно рассматривать как последовательность наборов $(\alpha_1(\boldsymbol{m}), \alpha_2(\boldsymbol{m}),\ldots,\alpha_{\boldsymbol{n}}(\boldsymbol{m}))$, или чисел в $\boldsymbol{k}$-ричной системе исчисления. Каждое из этих чисел $\in \boldsymbol{E}_{\boldsymbol{N}}$, где $\boldsymbol{N}=\boldsymbol{k}^{\boldsymbol{n}}$.

Тогда функцию $\boldsymbol{f}(\overline{X})$ из $P_{\aleph,k}$ можно рассматривать как функцию одной переменной из $E_{\aleph,N}$, принимающую значения из $E_{\aleph,k}$.

Функция $\boldsymbol{f}(\overline{X})$ из $P_{\aleph,N}$ называется детерминированной, если для любого $\boldsymbol{m}$ и для любых последовательностей $\overline{\alpha}$ и $\overline{\beta}$, таких, что $\overline{\alpha}\,(1)= \overline{\beta}\,(1),\ldots, \overline{\alpha}\,(\boldsymbol{m}) = \overline{\beta}\,(\boldsymbol{m})$ значения функции $\overline{\gamma}$ и $\overline{\delta}$: $\overline{\gamma}=\boldsymbol{f}(\overline{\alpha})$, $\overline{\delta}=\boldsymbol{f}(\overline{\beta})$ представляют собой последовательности, у которых также совпадают $\delta(1) = \gamma(1)$, $\delta(2) = \gamma(2)$, …., $\delta(\boldsymbol{m}) = \gamma(\boldsymbol{m})$. Таким образом, если в последовательностях аргументов функции совпадают первые $\boldsymbol{m}$ элементов, то в последовательностях результата первые $\boldsymbol{m}$ цифр также совпадают.

Рассмотренная ранее функция $f_1(\boldsymbol{x}) \in E_{\aleph,2}$ не является детерминированной.

Обозначим $\boldsymbol{P}_{д,\boldsymbol{k}}$ – множество всех детерминированных функций из $P_{\aleph,k}$. Детерминированная функция определяется последовательностью функций $\boldsymbol{k}$-значной логики, $\boldsymbol{f} = \{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_{\boldsymbol{m}}, \ldots\}$, где $\boldsymbol{f}_{\boldsymbol{m}}$ зависит от $\boldsymbol{mn}$ переменных:

$\boldsymbol{f}_1 = \boldsymbol{f}_1(\overline{X}(1))$,

$\boldsymbol{f}_2 = \boldsymbol{f}_2(\overline{X}(1), \overline{X}(2))$,

$\boldsymbol{f}_3 = \boldsymbol{f}_3(\overline{X}(1), \overline{X}(2), \overline{X}(3))$,…

Здесь

$\overline{X}(1) = (\boldsymbol{x}_1(1), \boldsymbol{x}_2(1), \ldots, \boldsymbol{x}_{\boldsymbol{n}}(1))$,

$\overline{X}(2) = (\boldsymbol{x}_1(2), \boldsymbol{x}_2(2), \ldots, \boldsymbol{x}_{\boldsymbol{n}}(2))$,

….

$\overline{X}(m) = (\boldsymbol{x}_1(\boldsymbol{m}), \mathrm{x}_2(\boldsymbol{m}), \ldots, \mathrm{x}_{\boldsymbol{n}}(\boldsymbol{m}))$.

Детерминированная функция $\boldsymbol{f}(\overline{X})$ может быть проинтерпретирована следующим образом: есть некоторый дискретный преобразователь (рис.1).

Рис.1. Дискретный преобразователь

Преобразователь имеет $\boldsymbol{n}$ входов, на которые подаются входные последовательности

$\{\alpha_1(1), \alpha_1(2), \ldots \alpha_1(\boldsymbol{m}), \ldots\}$,

$\{\alpha_2(1), \alpha_2(2), \ldots \alpha_2(\boldsymbol{m}), \ldots\}$,

…

$\{\alpha_n(1), \alpha_n(2),\ldots \alpha_n(\boldsymbol{m}),\ldots\}$,

в момент времени $\boldsymbol{i}$ на вход $\boldsymbol{x_j}$ подаётся $\alpha_j(\boldsymbol{i})$, т.е. всего в момент $\boldsymbol{i}$ на данный преобразователь подаётся вектор $(\alpha_1(\boldsymbol{i}), \alpha_2(\boldsymbol{i}),\ldots, \alpha_n(\boldsymbol{i}))$.

В момент времени 1 на входе $(\alpha_1(1), \alpha_2(1),\ldots, \alpha_n(1))$, в момент 2 – $(\alpha_1(2), \alpha_2(2),\ldots, \alpha_n(2))$, и так далее. В эти же моменты на выходе возникает выходная последовательность $\overline{\gamma}=\{\gamma(1), \gamma(2),\ldots, \gamma(\mathrm{m}),\ldots\}$. При этом $\gamma(n)=f(\overline{\alpha_1}, \overline{\alpha_2},...,\overline{\alpha_n})$. В таком преобразователе значение $\gamma(\boldsymbol{m})$ зависит только от входных последовательностей в моменты $\boldsymbol{t} = 1, 2, \ldots \boldsymbol{m}$, и не зависит от будущих значений входных последовательностей, поэтому получаемая с помощью дискретного преобразователя функция является детерминированной.

Поскольку детерминированная функция $\mathrm{F}(\overline{X_1}, \overline{X_2},..., \overline{X_n})$ полностью определяется последовательностью функций $\boldsymbol{k}$ –значной логики, получаем теорему [1]:

Теорема 3. Мощность множества всех детерминированных функций, зависящих от переменных $\overline{X_1}, \overline{X_2},..., \overline{X_n}$, равна $\aleph$.

Например, пусть $\boldsymbol{g}(\boldsymbol{x}_1, \boldsymbol{x}_2,\ldots \boldsymbol{x_n}) \in \boldsymbol{P_k}$. Тогда $\mathrm{F_g}(\overline{X_1}, \overline{X_2},..., \overline{X_n})=\{\boldsymbol{g}(\boldsymbol{x}_1(1), \boldsymbol{x}_2(1), \ldots, \boldsymbol{x_n}(1)), \boldsymbol{g}(\boldsymbol{x}_1(2), \boldsymbol{x}_2(2), \ldots, \boldsymbol{x_n}(2)), \ldots, \boldsymbol{g}(\boldsymbol{x}_1(\boldsymbol{m}), \mathrm{x}_2(\boldsymbol{m}), \ldots, \mathrm{x}_{\boldsymbol{n}}(\boldsymbol{m})),\ldots\}$ является детерминированной функцией, $\boldsymbol{F_g} \in \boldsymbol{P}_{\text{д},\boldsymbol{k}}$.

Если определить $\boldsymbol{g}(\boldsymbol{x}_1,\boldsymbol{x}_2)=\boldsymbol{x}_1\&\boldsymbol{x}_2$, то $\boldsymbol{F_g}(\boldsymbol{X}_1, \boldsymbol{X}_2)=\{ \boldsymbol{x}_1(1)\& \boldsymbol{x}_2(1), \boldsymbol{x}_1(2)\&\boldsymbol{x}_2(2),\ldots, \boldsymbol{x}_1(\boldsymbol{m})\&\boldsymbol{x}_2(\boldsymbol{m}), \ldots\}$

Обозначим $\widetilde{P_k}$ множество всех функций $\boldsymbol{F_g}$, таких, что $\boldsymbol{g} \in \boldsymbol{P_k}$.

Рассмотрим ещё один пример. Пусть $\boldsymbol{z}=\boldsymbol{x}+\boldsymbol{y}$ (суммирование чисел в системе исчисления с основанием k производится начиная с младшего разряда):

	……$\boldsymbol{x}(3)$	$\boldsymbol{x}(2)$	$\boldsymbol{x}(1)$
+	…….$\boldsymbol{y}(3)$	$\boldsymbol{y}(2)$	$\boldsymbol{y}(1)$
	…….$\boldsymbol{z}(3)$	$\boldsymbol{z}(2)$	$\boldsymbol{z}(1)$

Эта функция также принадлежит классу $\boldsymbol{P}_{д,\boldsymbol{k}}$. Старшие разряды суммы определяются соответствующими разрядами слагаемых и результатом сложения младших разрядов слагаемых, т.е., ***i***-й разряд суммы зависит только от ***j***-х (***j*≤*i***) разрядов слагаемых.

Однако ранее приведённая функция

$$f_1(\alpha)=\begin{cases}\{0,0,0,...\}, \text{ если } \alpha=\{0,0,0,...\},\\ \{1,1,1,...\}, \text{ если } \alpha\neq\{0,0,0,...\}\end{cases}$$ не принадлежит классу $\boldsymbol{P}_{д,2}$.

Задание детерминированных функций с помощью деревьев

Рассматриваем функции ***k***-значной логики от n переменных. Здесь ***k*** и ***n*** –целые числа. Обозначим $\boldsymbol{N=k^n}$.

Строим дерево с корнем, нулевой ярус вершин соответствует корню дерева. Из каждой вершины дерева исходит ***N*** дуг. Дуги считаем принадлежащими ярусу их концевой вершины, т.е. дуги из корня дерева – дуги первого яруса.

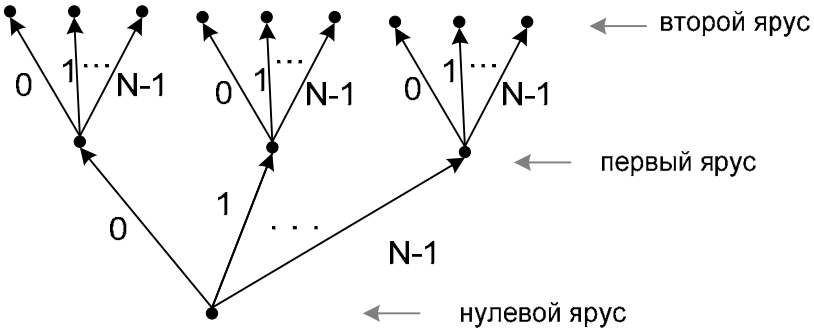


Рис. 2. Соответствие дуг дерева значениям аргументов функции.

Дуги каждого пучка нумеруем числами от 0 до ***N***-1 или их записями в ***k***-ричной системе исчисления (рис.2):

$$\underbrace{(0,0,...,0)}_{n},(0,0,...,0,1),...,(k-1,k-1,...,k-1).$$

Ветвь дерева – связное подмножество дуг, содержащее на каждом ярусе 1 дугу.

Ветви дерева сопоставляется последовательность $\alpha=\{\alpha(1), \alpha(2),\ldots\alpha(\boldsymbol{m}),\ldots\}$, где ***m*** – номер яруса, $\alpha(\boldsymbol{j})$ – номер дуги в *j*-м ярусе. $\alpha\in E_{\aleph,N}$. Таким образом, устанавливается взаимнооднозначное соответствие между ветвью дерева и последовательностью.

Пусть $\boldsymbol{f}(\overline{X})=\{\boldsymbol{f}_1(\boldsymbol{x}_1), \boldsymbol{f}_2(\boldsymbol{x}_1, \boldsymbol{x}_2),\ldots., \boldsymbol{f}_m(\boldsymbol{x}_1,\ldots \boldsymbol{x}_m),\ldots\}$. При помощи $\boldsymbol{f}(\overline{X})$ каждой дуге припишем число из $\boldsymbol{E}_k$. Рассмотрим дугу ***m***-го яруса. Для неё существует единственный путь из корня. Припишем этой дуге ***m***-й член последовательности – число $\gamma_m=\boldsymbol{f}_m(\alpha(1), \ldots \alpha(\boldsymbol{m}))$. Полученное дерево называется занумерованным деревом (деревом с занумерованными дугами).

Примеры.

1. Пусть $\boldsymbol{f}_{\&}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ – поэлементная конъюнкция. ***k***=2, ***N***=4. Этой функции соответствует занумерованное дерево, первые ярусы которого приведены на рис.3.

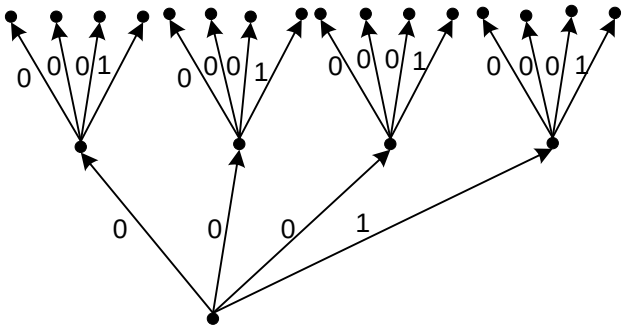


Рис. 3. Занумерованное дерево для конъюнкции

2. Пусть $z = f_+(x, y) = x + y$, при $k=2$ и $n=2$, тогда $N=4$. Определим элементы последовательности z:

$$z(m) = \begin{cases} x(m) + y(m) (\mathrm{mod}(2)), \text{при отсутствии переноса,} \\ x(m) + y(m) + 1 (\mathrm{mod}(2)), \text{при наличии переноса.} \end{cases}$$

На рис.4 представлены первые ярусы соответствующего занумерованного дерева. Для упрощения построения дерева вершины с единицей переноса помечены кружками.

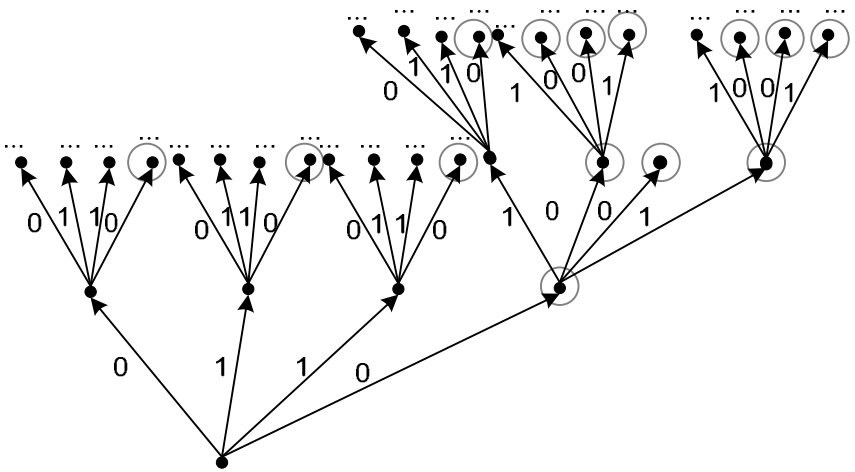


Рис. 4. Занумерованное дерево для сложения

Каждой детерминированной функции соответствует единственное дерево. Обратное, вообще говоря, неверно. Помеченное дерево может соответствовать более чем одной детерминированной функции, поскольку может существовать более одного решения уравнения $N=k^n$ (например, $k=N$, $n=1$). В случае же заданных k и n функция является единственной; $f(\overline{X}) = \{f_1(x_1), f_2(x_1, x_2), \ldots, f_m(x_1, \ldots x_m), \ldots\}$.

Вес детерминированной функции

Пусть β – вершина ***m***-го яруса занумерованного дерева. Совокупность всех ветвей, исходящих из вершины β, порождает некоторое дерево с корнем в β, которое называется специальным поддеревом. Оно определяется множеством всех последовательностей с фиксированным началом $\alpha(1),\alpha(2),\ldots, \alpha(\boldsymbol{m})$. Специальному поддереву с корнем в вершине β соответствует детерминированная функция $\boldsymbol{f}^{\beta}(\boldsymbol{X})=\{\boldsymbol{f}_1^{\beta}(\boldsymbol{x}_1), \boldsymbol{f}_2^{\beta}(\boldsymbol{x}_1, \boldsymbol{x}_2),\ldots\}$, при этом $\boldsymbol{f}_i^{\beta}(\boldsymbol{x}_1, \boldsymbol{x}_2,\ldots, \boldsymbol{x}_i) = \boldsymbol{f}_{m+i}(\alpha(1), \alpha(2),\ldots, \alpha(\boldsymbol{m}), \boldsymbol{x}_1, \boldsymbol{x}_2,\ldots, \boldsymbol{x}_i)$.

Два поддерева с корнями β и γ исходного дерева называются эквивалентными, если $\boldsymbol{f}^{\beta}(\boldsymbol{X})\equiv\boldsymbol{f}^{\gamma}(\boldsymbol{X})$.

Очевидно, что при естественном наложении двух эквивалентных поддеревьев их нумерации совпадают. Отношение эквивалентности поддеревьев является обычным отношением эквивалентности, т.е. это отношение рефлексивно, симметрично и транзитивно. Поэтому оно разбивает все поддеревья на классы эквивалентности.

Число ***r*** классов эквивалентности, на которое разбивается множество всех поддеревьев данного дерева, называется весом дерева, и соответственно, весом детерминированной функции.

Т.е. вес функции – максимальное число попарно неэквивалентных деревьев в занумерованном дереве этой функции. Оно может быть и бесконечно для детерминированной функции.

Если рассмотреть дерево для конъюнкции (см. рис. 3), то легко видеть, что все поддеревья этого дерева эквивалентны, $\boldsymbol{r}=1$.

Вес дерева для суммы, которое представлено на рис.4, $\boldsymbol{r}=2$.

Если рассмотреть дерево для функции $\boldsymbol{x}^2$, $\boldsymbol{k}=2$, $\boldsymbol{n}=1$, $\boldsymbol{N}=2$, то легко видеть, что, например, все деревья на левой ветви не эквивалентны, так как $\left\{\underbrace{0\ldots0}_{i}\alpha(i+1)\ldots\right\}^2 = \left\{\underbrace{0\ldots0}_{2i}\alpha(i+1)\ldots\right\}$.

Для занумерованных деревьев можно ввести нумерацию вершин. Перенумеруем классы эквивалентности специальных подде-

ревьев числами 0, 1,… так, чтобы у всего дерева был класс 0. Вершине присваивается номер класса дерева с корнем в этой вершине.

Нумерация вершин нулевого и первого яруса дерева для конъюнкции представлена на рис.5. Поскольку все поддеревья эквивалентны, все вершины имеют номер 0.

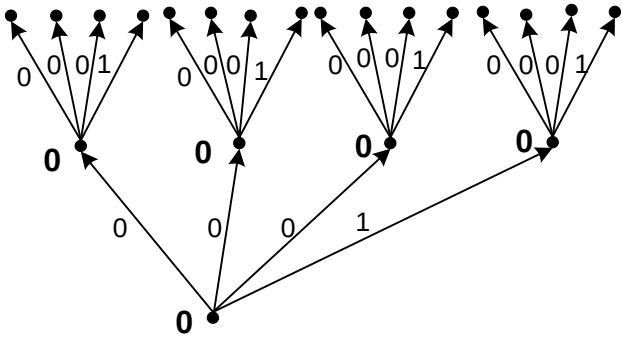


Рис. 5. Нумерация вершин дерева для конъюнкции

Нумерация вершин нулевого, первого и второго яруса дерева для сложения в двоичной системе представлена на рис.6.

Рассмотрим произвольную ветвь в дереве с занумерованными вершинами и дугами, пусть она проходит через вершины β_0, β_1,…, β_i,… β_j,… с номерами χ_0, χ_1,…, χ_i,…χ_j,… Пусть в этой ветви совпадают номера вершин i и j при $i<j$ (в силу ограниченности веса функции такие вершины найдутся). Выбираем в данной ветви для всех пар i, j, $i < j$, таких, что $\chi_i = \chi_j$, наименьший номер j. Произведём усечение ветви, сохранив начальный отрезок до β_j. Повторив процедуру для всех ветвей дерева, получим усечённое дерево. Усечённое дерево для конъюнкции представлено на рис.7, для суммы – на рис.8.

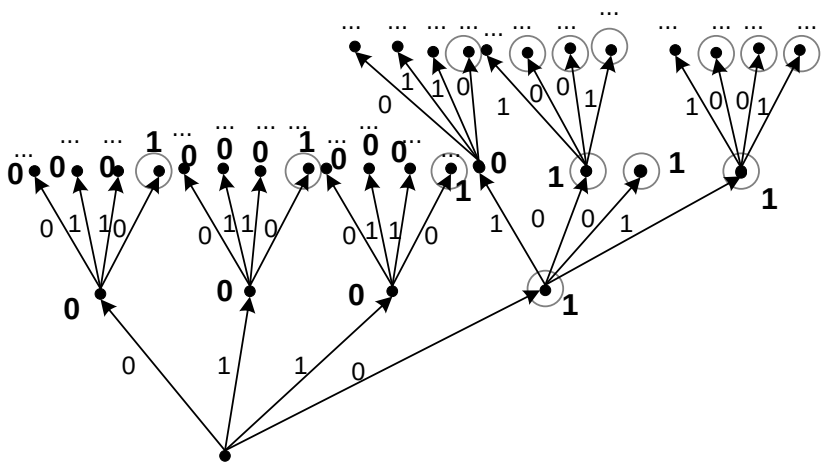


Рис. 6. Нумерация вершин дерева для сложения

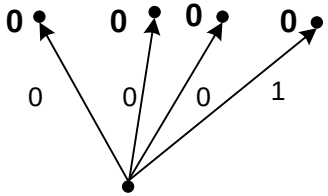


Рис.7. Усечённое дерево для конъюнкции

Максимальная длина ветви усечённого дерева для сложения равна двум, в дереве для конъюнкции все ветви имеют длину 1.

Для случая конечного веса r справедливо $j \le r$ и усечённое дерево конечно.

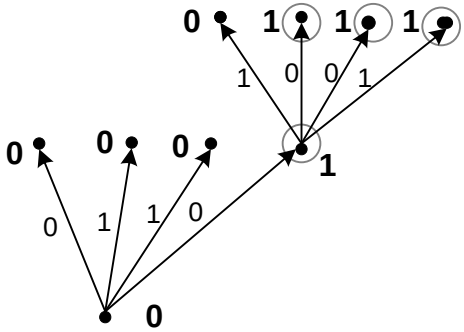


Рис. 8. Усечённое дерево для суммы

Ограниченно-детерминированные функции и способы их задания

Детерминированная функция $f(x_1, x_2, \ldots, x_n)$ называется ограниченно-детерминированной, если она имеет конечный вес.

Класс всех ограниченно-детерминированных функций, принадлежащих классу $P_{д,k}$, обозначается $P_{од,k}$.

Диаграммы для детерминированных функций

Для каждой функции, принадлежащей классу $P_{од,k}$, можно построить занумерованное дерево. Усечённое дерево для ограниченно-детерминированной функции имеет конечное число вершин. Отождествив в усечённом дереве вершины с одинаковыми номерами, получаем диаграмму Мура. Поскольку при таком отождествлении теряется порядок дуг дерева, каждая дуга из вершины β, соответствующая аргументу α, помечается парой $(\alpha, f^{\beta}(\alpha))$.

Если ограниченно-детерминированная функция имеет вес r, то диаграмма Мура имеет r вершин, причём одна из них выделена в качестве начальной вершины. Из каждой вершины исходит $N=k^n$ дуг, дугам приписаны пары $(0,\gamma^{(0)}),\ldots, (N-1, ,\gamma^{(N-1)})$. Диаграммы, обладающие такими свойствами, называются диаграммами Мура.

Например, диаграмма для конъюнкции имеет только одну вершину. Она представлена на рис.9.

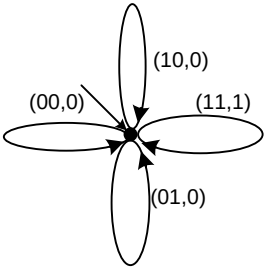


Рис.9. Диаграмма Мура для поразрядной конъюнкции

Усечённое дерево для сложения содержит два типа вершин. Соответствующая ему диаграмма Мура представлена на рис. 10.

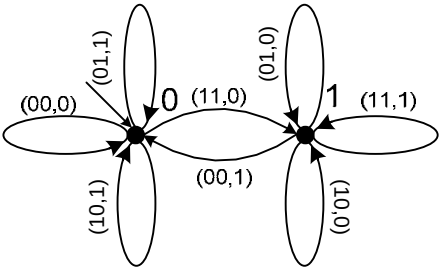


Рис.10. Диаграмма Мура для двоичного сложения

Диаграммы Мура позволяют задавать ограниченно-детерминированные функции любого конечного веса ***r***.

По диаграмме ограниченно-детерминированная функция восстанавливается однозначно, но по одной ограниченно-детерминированной функции можно построить и более одной диаграммы, что будет продемонстрировано позднее.

Теорема 4. Число $p(k,n,r)$ ограниченно-детерминированных функций из $P_{\aleph,k}$, зависящих от $x_1, x_2,\ldots, x_n$ и имеющих вес r, не более, чем $(rk)^{rk^n}$.

Доказательство. Из каждой вершины исходит k^n дуг, i-я дуга соединена с одной из r вершин, ей приписана пара (α_i, γ), где $0 \le \gamma \le k-1$.

В диаграмме r вершин, занумерованных числами от 0 до r-1 (0 – выделенная, начальная вершина), из каждой вершины выходят k^n дуг, занумерованных числами от 0 до N-1. Всего в диаграмме $r\mathbf{N}$ дуг, каждая дуга может быть соединена с каждой из r вершин и ей может быть приписано каждое из k чисел:

$p(k,n,r) \le (rk)^{rN} \le (rk)^{rk^n}$.

Вопросы и упражнения

1. Какому набору аргументов соответствует строка 11 таблицы функции $f(x_1,x_2)$ при k=4? Сколько всего строк в таблице?
2. Выполняется ли закон Де-Моргана для функции $\bar{x} = x + 1 (\mathrm{mod}\, k)$?
3. Выполняется ли закон ассоциативности для функции max?
4. Каков вес функции $x\&2$ при k=3?
5. Каков вес функции $x+5$ при k=4?
6. Каков вес функции $x \times 2$ при k=3?
7. Функция является ограниченно-детерминированной. Каково может быть число вершин m в сокращённом дереве для заданного $N=k^n$?
8. Вес функции равен 3 при k=2. Оценить число вершин в сокращённом дереве.
9. Пусть в таблице функции 15 строк. Каковы могут быть значения n и k? Если в таблице 16 строк?
10. Сколько ветвей в дереве на первом уровне при n=3, k=2?
11. Всегда ли детерминированная функция имеет конечный вес?

12. Как отражается коммутативность функции на виде дерева при $k= n=2$? На диаграмме?

13. В каком случае дерево соответствует более чем одной функции?

14. Может ли дерево соответствовать более чем одной функции при $N=5$?

15. Почему нельзя задать деревом недетерминированную функцию?

16. При каком условии диаграмма Мура конечна?

17. Как связан вес функции с числом вершин диаграммы Мура?

ГЛАВА 2. ОСНОВНЫЕ ТИПЫ ПРЕОБРАЗУЮЩИХ АВТОМАТОВ

Глава посвящена описанию различных типов преобразующих автоматов, соответствию между различными типами автоматов, а также методам минимизации автоматов.

Автомат Мили

Конечный автомат (автомат Мили) $S=< V_a, Q, V_b, q_0, F, G>$, где

$V_a= \{a_1, a_2,\ldots, a_m\}$, $m\geq 1$ – входной алфавит автомата,

$V_b= \{b_1, b_2, \ldots, b_n\}$, $n\geq 1$ – выходной алфавит автомата,

$Q= \{q_0, q_1,\ldots, q_k\}$, $k\geq 0$ – внутренний алфавит (алфавит состояний),

$q_0 \in Q$ – начальное состояние автомата,

$F: Q\times V_a \rightarrow Q$ – функция переходов,

$G : Q\times V_a \rightarrow V_b$– функция выходов.

Автомат однозначно задает отображение $V_a^* \rightarrow V_b^*$ (входной цепочки символов в выходную цепочку).

Здесь рассматриваются инициальные автоматы, где входное состояние задано статически, в ряде случаев начальное состояние рассматривается как функция дискретного времени $q_0= q(0)$.

Например, задан автомат $S_1 = < V_a, Q, V_b, q_0, F, G >$:
здесь $V_a = V_b = \{a, b\}$, $Q = \{A, B\}$, $q_0 = A$. Функции переходов и выходов могут быть заданы в непосредственно для всех наборов аргументов:

$f(A, a) = A$; $g(A, a) = a$;
$f(A, b) = B$; $g(A, b) = a$;
$f(B, a) = A$; $g(B, a) = b$;
$f(B, b) = B$; $g(B, b) = b$.

Эти функции также могут быть заданы в виде двух отдельных таблиц (таблицы переходов и таблицы выходов) или в виде объединенной таблицы переходов-выходов (другое название этой таблицы – автоматная таблица), в которой по столбцам указаны исходные состояния, по строкам – входы. В ячейке, соответствующей состоянию q_i и входному символу a_j, через запятую указываются состояние, в которое переходит автомат ($f(q_i, a_j)$) и соответствующий выходной символ ($g(q_i, a_j)$). Для случая двух таблиц (таблицы переходов и таблицы выходов) содержимое ячеек очевидно.

Объединённая таблица входов-выходов для автомата S_1 имеет вид:

	A	B
a	A, a	A, b
b	B, a	B, b

Для многих случаев удобной формой представления автомата является граф переходов автомата.

Граф переходов автомата определяется следующим образом:

множество вершин графа: каждому элементу множества Q соответствует вершина;

множество дуг определяется отображениями F и G, причём F определяет связи (переходы состояний), а G – выходы. Если $f(q_i, a_j) = q_k$ и $g(q_i, a_j) = b_s$, то на диаграмме рисуется дуга из q_i в q_k с пометкой (a_j, b_s) (входной символ, выходной символ). Начальное состояние на графе обозначается стрелкой (не исходящей из других вершин), ведущей в соответствующую начальному состоянию вершину (рис 11).

Диаграмма (граф переходов автомата), представляющая автомат S_1, изображена на рис. 11.

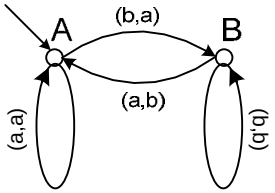


Рис. 11. Граф переходов автомата S_1

По графу переходов автомата несложно восстановить его полное описание.

Модель автомата – абстрактное устройство с входной и выходной лентами и управляющей головкой. В каждый момент времени автомат находится в одном из состояний множества Q (например, q_i), управляющая головка находится над одной из ячеек полубесконечной входной ленты. Автомат воспринимает символ входного алфавита, содержащийся в обозреваемой ячейке (например, a_j) входной ленты, и печатает один символ выходного алфавита ($b_s=g(q_i,a_j)$) на выходной ленте и переходит в состояние, определяемое функцией перехода (в нашем примере в $q_k=f(q_i,a_j)$). Затем автомат переходит к следующим ячейкам входной и выходной лент. Время в данной абстрактной модели считаем дискретным.

Существуют две традиции в задании автоматов. В предшествующем описании автомата время явным образом не задавалось. В описании автомата может использоваться явное задание дискретного времени, т.е. номера такта t, $t \in E_{\aleph_0}$ ($E_{\aleph_0}$ = {0,1,2,…}), $a(t) \in V_a$, $b(t) \in V_b$, $q(t) \in Q$.

Тогда работа автомата описывается с помощью рекуррентных соотношений:

$$\begin{cases} q(t+1) = f(q(t), a(t)), \\ b(t) = g(q(t), a(t)). \end{cases}$$

Иногда рассматривается $b(t+1)=g(q(t), a(t))$ – автомат с задержкой; далее такие автоматы не рассматриваются.

Отображения ***F*** и ***G*** в общем случае частичные, что описано в следующих разделах.

Пример. Пусть граф переходов автомата S_2 представлен на рис.12. Рассмотрим пример получения выходной цепочки по входной цепочке ***aabb***. Вначале автомат находится в состоянии q_0. При получении входного символа ***a*** автомат переходит в состояние q_1 и выдаёт символ ***c***. Затем, при входном символе ***a*** автомат, оставаясь в состоянии q_1, выдает символ ***d***, и так далее. Всего по входной цепочке ***aabb*** получается выходная цепочка ***cdcc***.

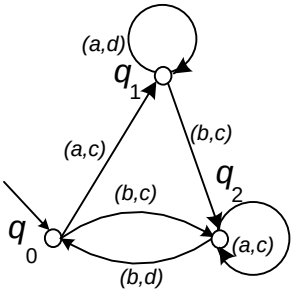


Рис.12. Граф переходов автомата $\mathbf{S_2}$

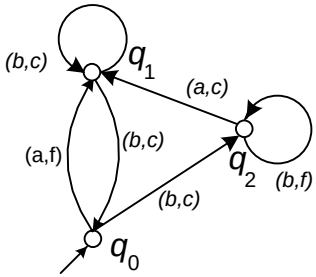


Рис.13. Граф переходов автомата $\mathbf{S_3}$

Легко видеть, что построенные графы переходов автоматов Мили соответствуют диаграммам Мура, полученным для некоторых ограниченно-детерминированных функций. Таким образом, если рассматривать диаграмму Мура как граф переходов некоторого автомата, то по входной цепочке (аргументу соответствующей ограниченно-детерминированной функции) будет получена выходная цепочка (результат вычисления функции при заданном аргументе).

Функции переходов и выходов автомата Мили определяют автоматное отображение $V_a^* \rightarrow V_b^*$. При этом для любой входной цепочки $\alpha = a_1 a_2 \ldots a_k$ значение $f(q_0, a_1 a_2 \ldots a_k) = f(\ldots f(f(q_0, a_1), a_2), \ldots a_k)$.

Дадим эквивалентное индуктивное определение функции переходов:

1. $f(q_i, a_j)$ определяется по таблице перехода автомата,

2. $f(q_i, \alpha\, a_j) = f(f(q_i, \alpha), a_j)$.

Соответствующая функция выхода:

1. $g(q_i, a_j)$ определяется по таблице перехода автомата,

2. $g(q_i, \alpha\, a_j) = g(f(q_i, \alpha), a_j)$.

Тогда входному слову $\alpha = a_1 a_2 \ldots a_k$ ставится в соответствие выходное слово $\omega(\alpha) = g(q_0, a_1)\, g(q_0, a_1\, a_2) \ldots g(q_0, a_1 a_2 \ldots a_k)$.

Это отображение, ставящее в соответствие входным словам выходные слова, называется автоматным отображением, или автоматным оператором, реализуемым автоматом S. Автоматный оператор обозначается: $S(\alpha) = \omega$.

Автоматное отображение можно определить индуктивно, как и функцию переходов:

1. $S(q_i, a_j) = g(q_i, a_j)$,

2. $S(q_i, \alpha\, a_j) = S(q_i, \alpha)\, g(f(q_i, \alpha), a_j)$.

Как и ранее, длина цепочки α обозначается $|\alpha|$.

Свойства автоматного отображения ($S(\alpha)$ рассматриваем как $S(q_0, \alpha)$):

1. α и $\omega = S(\alpha)$ имеют одинаковую длину $|\alpha| = |S(\alpha)|$.

2. Если $\alpha = \alpha_1 \alpha_2$, и $S(\alpha_1 \alpha_2) = \omega_1 \omega_2$, где $|\alpha_1| = |\omega_1|$, то $S(\alpha_1) = \omega_1$.

То есть автоматный оператор является оператором без «предвосхищения», без заглядывания вперед, например, невозможно построить инверсию слова.

Определим, что состояние q_j достижимо из состояния q_i, если $\exists \alpha \in V_a^*$, такое, что $f(q_i, \alpha) = q_j$.

Автомат S называется сильно связным, если любое состояние достижимо из любого другого. Автоматы S_1 и S_2 являются сильно связными.

Два автомата S и T изоморфны, если входы, выходы и состояния автомата S можно переименовать таким образом, чтобы таблица переходов автомата S превратилась в таблицу переходов автомата T.

Например, автомат S_2 на рис. 12 изоморфен автомату S_3 на рис. 13.

Соответствия между параметрами автоматов (параметру автомата S_2 ставим в соответствие параметр автомата S_3): входов: $a - b$, $b - a$; выходов: $c - c$, $d - f$; состояний: $q_0 - q_0$, $q_1 - q_2$, $q_2 - q_1$.

Пусть T и S – автоматы с одинаковыми входными и выходными алфавитами. Состояние q автомата S и состояние r автомата T называются неразличимыми, если для любой входной цепочки $\alpha \in V_a^*$ выполнено $S(q,\alpha)=T(r,\alpha)$. При $T=S$ говорят о неразличимых состояниях одного автомата.

Если неразличимы входные состояния двух автоматов, то они реализуют одно и то же автоматное отображение. В этом случае автоматы эквивалентны.

Переход от автомата к эквивалентному – эквивалентное преобразование автомата. Одним из таких преобразований, практически полезным, является минимизация числа состояний автомата.

Метод Хафмена минимизации числа состояний автомата

Этот метод работает как для автоматов Мили, так и других типов автоматов (что будет показано позднее).

Идея метода – объединение в один класс предположительно эквивалентных состояний, а затем проверка, являются ли состояния действительно эквивалентными (неразличимыми). Затем, если состояния не эквивалентны, классы разбиваем на подклассы, и проверка повторяется для построенных классов. Если состояния в полученных классах ведут себя одинаково, то процедура закончена и получившиеся классы соответствуют состояниям минимизированного автомата.

Например, рассмотрим автомат S_D, граф переходов которого представлен на рис. 14. Изложение алгоритма будет сопровождаться рассмотрением процесса минимизации для этого автомата.

Алгоритм минимизации автомата

1. Составляем таблицы переходов и выходов. По строкам указываются входы, по столбцам – состояния, для большей наглядности в первом подстолбце каждого столбца указаны переходы, во втором – соответствующие выходы.

	A		***B***		***C***		***D***		***E***		***F***		***G***	
0	***B***	**1**	***D***	**1**	***E***	**0**	***D***	**1**	***A***	**0**	***G***	**0**	***D***	**0**
1	***F***	**0**	***C***	**0**	***G***	**1**	***C***	**0**	***F***	**1**	***E***	**1**	***C***	**1**

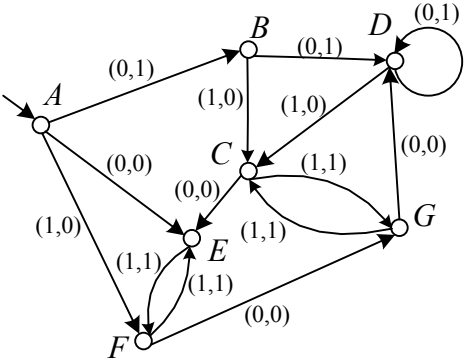


Рис.14. Граф переходов автомата S_D

2. Составляем классы предположительно эквивалентных состояний (предположительно эквивалентными являются состояния, при переходе из которых при одинаковых входах выдаются одинаковые выходы, т.е. для данного представления таблицы переходов это те состояния, у которых одинаковы вторые подстолбцы).

№ класса	**1**	**1**	**2**	**1**	**2**	**2**	**2**
Вход\состояние	***A***	***B***	***C***	***D***	***E***	***F***	***G***
0	K_1	K_1	K_2	K_1	K_1	K_2	K_1
1	K_2	K_2	K_2	K_2	K_2	K_2	K_2

В данном случае это классы $K_1 = \{A, B, D\}$, $K_2 = \{C, E, F, G\}$.

3. Составляем таблицу переходов, в которой вместо состояний указываются классы, которым принадлежат состояния. Для наглядности в таблицу добавлена строка, с указанием для каждого состояния номера класса, которому оно принадлежит.

Если в пределах одного класса состояния ведут себя по-разному, то класс разбивается на подклассы, включающие состояния, ведущие себя одинаково (т.е. имеющие одинаковые столбцы в построенной таблице), и возвращаемся к шагу 3 (в рассматриваемом случае классы разбиваются, таблицы приводятся после алгоритма).

4. Если во всех классах состояния ведут себя одинаково, то это действительно классы эквивалентности. Строится автомат, состояниями которого являются классы эквивалентности исходного автомата.

В нашем примере класс K_2 разбивается на два класса: $K_3 = \{C, F\}$ и $K_4 = \{E, G\}$.

№ класса	1	1	2	1	2	2	2
Вход\состояние	*A*	*B*	*C*	*D*	*E*	*F*	*G*
0	K_1	K_1	K_4	K_1	K_1	K_4	K_1
1	K_3	K_3	K_4	K_3	K_3	K_4	K_3

После этого разбиения состояния в классах действительно эквиваленты. Граф переходов полученного автомата S_D' представлен на рис.15. Этот автомат эквивалентен исходному автомату S_D.

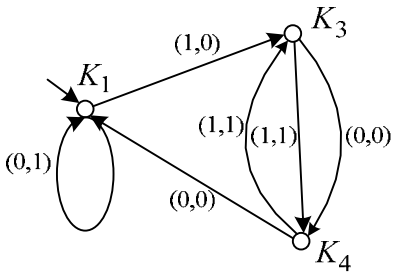


Рис.15. Граф переходов автомата S_D'

Автоматы Мура

Рассмотрим ещё один тип автоматов – автоматы Мура. Эти автоматы естественным образом возникают в ряде приложений. Автоматы Мура отличаются от автоматов Мили тем, что здесь одному состоянию (а не переходу) соответствует один выход.

Конечный автомат Мура: $S=< V_a, Q, V_b, q_0, F, G>$, где

$V_a=\{a_1, a_2,\ldots, a_m\}$, $m\geq 1$ – входной алфавит автомата,

$V_b= \{b_1, b_2, \ldots, b_n\}$, $n\geq 1$ – выходной алфавит автомата,

$Q = \{q_0, q_1,\ldots, q_k\}$, $k\geq 0$ – внутренний алфавит (алфавит состояний),

$q_0 \in Q$ – начальное состояние автомата (рассматриваем, как и ранее, инициальный автомат, иногда рассматривают начальное состояние как функцию времени $q_0= q(0)$),

F – функция переходов; $F: Q\times V_a \rightarrow Q$,

G – функция выходов; $G: Q \rightarrow V_b$.

Приняты две схемы задания автоматов Мура:

Первая схема

$$\begin{cases} q(t+1) = f(q(t), a(t)), \\ b(t)= g(q(t)). \end{cases}$$

Вторая схема

$$\begin{cases} q(t+1) = f(q(t), a(t)), \\ b(t)= g(q(t+1)). \end{cases}$$

При работе по первой схеме выход автомата однозначно соответствует состоянию, из которого совершается переход, по второй – состоянию, в которое автомат переходит. Хотя при записи уравнений первая схема выглядит более естественно, позднее будет показано, что в некоторых случаях вторая схема обеспечивает большую универсальность работы: при работе по первой схеме выходной символ на данном такте не зависит от входного символа данного такта. Это оказывается не всегда удобно, хотя в ряде приложений используется именно это свойство автоматов Мура, работающих по первой схеме. Кроме того, любой автомат Мура, работающий по второй схеме, можно построить таким образом, что он будет работать подобно автомату, работающему по первой схеме, т.е.

автомат Мура, работающий по второй схеме, является более универсальным.

Все отображения автомата Мура могут быть заданы в функциональной форме и в виде таблиц. Автомат Мура, так же как и автомат Мили, может быть представлен с помощью графа переходов. Граф переходов автомата Мура определяется следующим образом:

множество вершин графа: каждому элементу множества $\boldsymbol{Q}$ соответствует вершина;

начальное состояние на графе обозначается стрелкой, ведущей в это состояние (так же, как для автомата Мили);

множество ребер определяется отображением $\boldsymbol{F}$: если $\boldsymbol{f(q_i,a_j)=q_k}$, то на диаграмме рисуется дуга из $\boldsymbol{q_i}$ в $\boldsymbol{q_k}$ с пометкой $\boldsymbol{a_j}$ (входной символ для перехода);

функция $\boldsymbol{G}$ определяет метки у состояний: если $\boldsymbol{g(q_i)=b_s}$, то состояние $\boldsymbol{q_i}$ помечается символом $\boldsymbol{b_s}$.

Например, рассмотрим автомат Мура $\boldsymbol{S_4}$, граф переходов которого представлен на рис. 16.

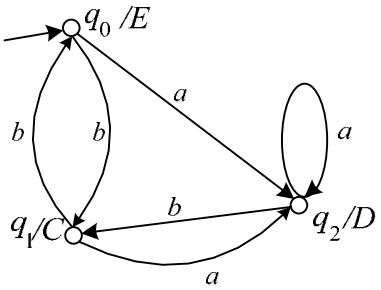


Рис.16. Граф переходов автомата Мура $\mathbf{S_4}$

Здесь выходы, соответствующие состояниям, изображены справа от состояния, выходной символ отделён от обозначения состояния слешем. Если рассматривать работу автомата по первой схеме, то входу ***aabb*** будет соответствовать выход ***EDDC***, если же по второй схеме, то этому же входу соответствует выход ***DDCE***.

Рассмотрим возможность построения для автомата Мура эквивалентного автомата Мили.

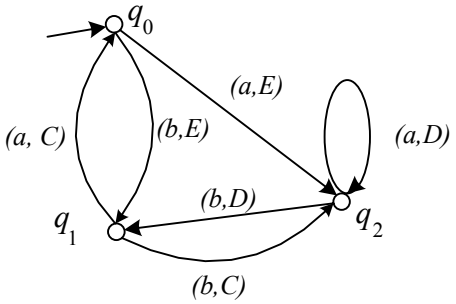


Рис.17. Автомат Мили, построенный по автомату $\boldsymbol{S_4}$
при работе по первой схеме

При рассмотрении первой схемы работы автомата Мили выходные сигналы, соответствующие любому состоянию, переносятся на дуги, выходящие из вершины графа, представляющей данное состояние. При рассмотрении второй схемы работы автомата Мура выходной символ, приписанный состоянию, переносится на дуги, входящие в вершину, соответствующую этому состоянию. Автомат Мили, эквивалентный автомату Мура $\boldsymbol{S_4}$, представленному на рис. 16, при работе по первой схеме, дан на рис. 17 (все дуги, ведущие из состояния, имеют одинаковые выходы), а при работе по второй схеме – на рис.18 (все дуги, ведущие в состояние, имеют одинаковые выходы). Таким образом, по автомату Мура всегда можно построить автомат Мили.

По выразительной мощности эти модели (автоматы Мили и Мура) эквивалентны, если используется вторая схема для представления автоматов Мура.

Для построения автомата Мура (вторая схема) по автомату Мили используется граф переходов автомата Мили.

1. Если все дуги, ведущие в некоторое состояние q_k, имеют одинаковые выходные пометки b_s, то эта пометка просто переносится на это состояние (рис.19).

Формально это условие для состояния q_k и выхода b_s можно записать так:

$$\forall q_i,q_j,a_n,a_m\ (f(q_i,a_n) = f(q_j,a_m) = q_k \Rightarrow g(q_i,a_n) = g(q_j,a_m) = b_s).$$

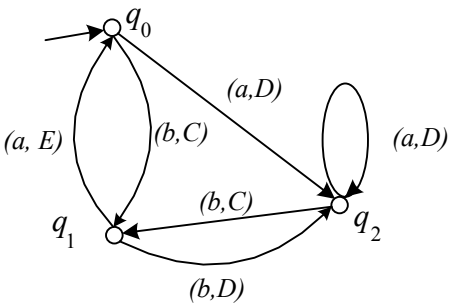


Рис.18. Автомат Мили, построенный по автомату S_4
при работе по второй схеме

2. Общий случай: в некоторые вершины ведут дуги, помеченные разными выходными символами. В этом случае все такие вершины q_k расщепляются на множество вершин (расщепление состояния), помечаемых символами $<q_k, b_j>$ (рис.20). Условная запись расщепления состояний: $q_k \rightarrow \{<q_k, b_j> \mid f(q_i, a)=q_k, g(q_i, a)=b_j\}$.

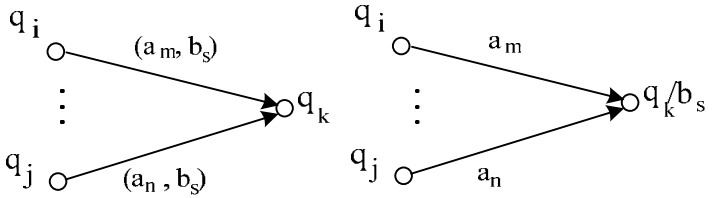


Рис. 19. Условное изображение перенесения выходного символа на состояние

В общем случае число состояний автомата может увеличиваться. Затем для каждого состояния q_i исходного автомата Мили и для каждой дуги из состояния q_i в состояние q_k с пометкой (a_p,b_s) (a_p – вход, b_s – выход) строятся дуги из всех $<q_i,b_j>$ в $<q_k, b_s>$ и помечаются a_p.

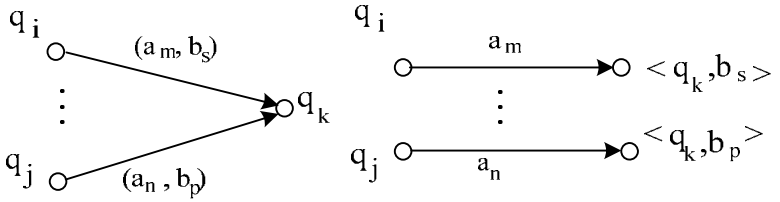


Рис. 20. Условное изображение расщепления состояний

Кроме того, если начальное состояние q_0 расщепилось, вводится новое начальное состояние, в которое не ведет ни одна дуга.

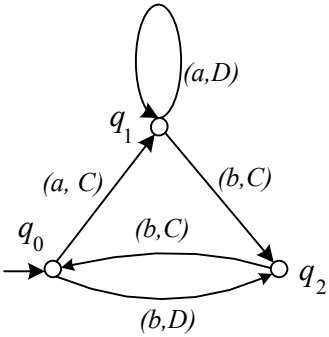
Затем приписываем состояниям соответствующие выходы и переобозначаем состояния.

Например, рассмотрим автомат Мили S_5 (рис. 21, а). У этого автомата состояния q_1 и q_2 расщепляются (для состояния q_1 на входящих дугах имеются выходные символы C и D , а для состояния q_1 на входящих дугах выходные символы также C и D), а q_0 не расщепляется (на входящей дуге единственный выходной символ C), поэтому новое входное состояние не вводится. Полученный после расщепления состояний эквивалентный автомат Мура приведён на рис. 21,б. Тот же автомат после переобозначения состояний приведен на рис. 21,в.

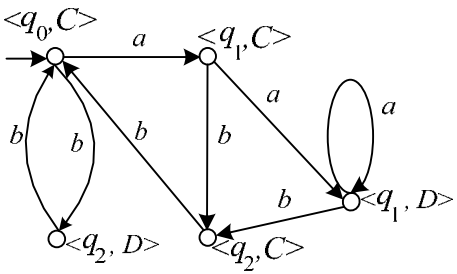
Полный алгоритм построения эквивалентного автомата Мура по автомату Мили выглядит следующим образом:

1) расщепляем состояния;
2) вводим дополнительное входное состояние (если входное состояние расщепилось);
3) строим дуги, соответствующие дугам исходного автомата;
4) переносим выходные символы на соответствующие состояния;

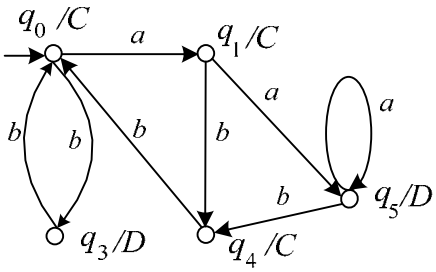
5) переобозначаем состояния.



а)



б)



в)

Рис. 21. Построение по автомату Мили эквивалентного автомата Мура (вторая схема задания)

Таким образом, установлено соответствие между автоматами Мили и Мура. По автомату Мура всегда можно построить эквивалентный автомат Мили. Способ построения зависит от того, по какой схеме работает автомат Мура. По автомату Мили можно построить автомат Мура, работающий по второй схеме. Необходимость использования второй схемы при этом построении обусловлена следующим обстоятельством. Для того, чтобы по автомату Мили можно было построить автомат Мура по первой схеме, в соответствующем автомате Мили должны быть одинаковые выходы на всех дугах, **выходящих из** состояния. Для того, чтобы по автомату Мили построить автомат Мура, работающий по второй схеме, одинаковые выходы должны быть на дугах, **ведущих в** состояние. Первое условие может выполняться, а может и не выполняться (изменить эту ситуацию невозможно), выполнения второго условия можно достичь с помощью эквивалентного преобразования автомата.

Частичные автоматы

Автомат S называется частичным автоматом, если хотя бы одна из двух функций (F или G) не полностью определена, т.е. для некоторых пар (состояние, вход) значение F или G не определено (обозначается прочерками в таблице).

Состояния A и B называются псевдоэквивалентными (обозначается $A \cong B$), если для них одновременно не определены или определены и равны функции F и G. Эти функции для частичных автоматов определяются следующим образом:

Функция $f(q_i,\alpha)$**:**

1) $f(q_i, a_j)$ – по таблице переходов автомата,

2) если $f(q_i, \alpha)$ определена, то $f(q_i, \alpha\, a_j) \cong f(f(q_i, \alpha), a_j)$,

3) если $f(q_i,\alpha)$ не определена, то не определена и $f(q_i, \alpha\, a_j)$ для всех a_j.

Функция $g(q_i, \alpha\, a_j)$**:**

1) $g(q_i, a_j)$ – по таблице автомата,

2) $g(q_i, \alpha\, a_j) \cong g\,(f(q_i, \alpha), a_j)$.

Автоматное отображение:

1) $S(q_i, a_j) = g(q_i, a_j)$ (если $g(q_i, a_j)$ не определена, то $S(q_i, a_j)$ считаем равным прочерку),

2) если $f(q_i, \alpha)$ определена, то $S(q_i,\alpha\, a_j)=S(q_i,\alpha)\, g(f(q_i, \alpha)\, a_j)$. Если $g(f(q_i, \alpha), a_j)$ не определена, то считаем её равной прочерку),

3) если $f(q_i, \alpha)$ не определена, то не определено и $S(q_i,\alpha\, a_j)$ для всех a_j.

Входное слово α, для которого $S(q_0,\alpha)$ определено, называется допустимым для S.

Отметим неравноправность функций f и g – неопределенность g не препятствует допустимости слова, а неопределенность функции f для некоторого слова означает так же недопустимость любого его продолжения.

Состояния $q_i \in S$ и $r_j \in T$ псевдонеотличимы (псевдоэквивалентны), если для любой цепочки α $S(q_i, \alpha) \cong T(r_j, \alpha)$ (т.е. области определённости и неопределённости для q_i и r_j совпадают, и в области определённости отображения совпадают).

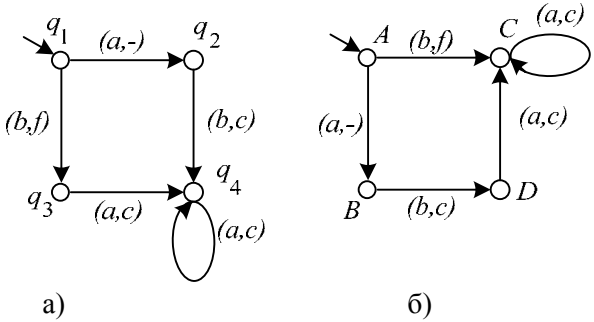
Автоматы S и T псевдонеотличимы, если для любого состояния автомата S найдётся псевдонеотличимое от него состояние автомата T, и, наоборот, для любого состояния автомата T найдётся псевдонеотличимое от него состояние автомата S.

Для полностью определённых автоматов псевдонеотличимость совпадает с обычной неотличимостью.

Отношение псевдонеотличимости является отношением эквивалентности.

Пример псевдоэквивалентных S_6 и S_7 автоматов приведён на рис.22,а ,б.

Состояниям первого автомата соответствуют состояния второго автомата: $q_1 – A$, $q_2 – B$, $q_3 – C$ и D, $q_4 – C$, D; соответствия состояниям второго автомата: $A – q_1$, $B – q_2$, $C – q_3$ и q_4, $D – q_3$ и q_4.



а) б)

Рис.22. Пример псевдоэквивалентных автоматов

Недостатком введённого определения псевдоэквивалентности является необходимость совпадения областей определения автоматного отображения. Можно дать определение, не требующее проверки совпадения областей: автомат доопределяется введением нового, дополнительного состояния и переходов в это состояние. Для каждого состояния, не имеющего переходов при данном входном символе a_i, строится переход во введённое дополнительное состояние, входной символ для этого перехода – a_i, выходной символ перехода – прочерк. Для введённого дополнительного состояния переход по любому входу осуществляется в это же состояние, с выходом, равным прочерку. Прочерк в функции выхода рассматривается как дополнительная буква. Таким образом, автомат становится полностью определённым. При этом вновь введённое состояние не эквивалентно в доопределённом автомате никакому из состояний исходного автомата.

Например, при доопределении автомата на рис. 22, а, получается автомат, представленный на рис. 23. Введено дополнительное состояние S_5. Построены дополнительные переходы в состояние S_5 из состояния S_2 при входном символе a, и из состояний S_3 и S_4 при входном символе b, а также переходы из состояния S_5 в это же состояние при всех входах.

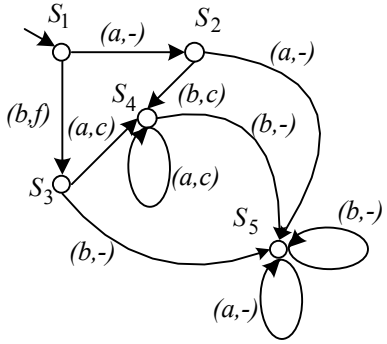


Рис. 23. Пример доопределения не полностью определённого автомата

Минимизация не полностью определённых автоматов возможна двух типов: «строгая», с сохранением областей определения (что производится просто, как для доопределённого автомата), и через покрытия состояний, когда требуется совпадение переходов только в области определённости [3], последняя здесь не рассматривается.

Если проводить строгую минимизацию, то автомат на рис. 22, а минимизируется до автомата, граф переходов которого приведен на рис. 24.

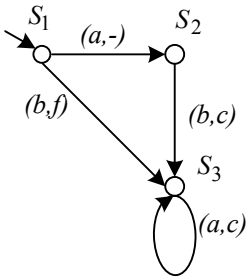


Рис. 24. Пример минимизации не полностью определённого автомата

Вопросы и упражнения

1.Найти автоматное отображение для цепочки aabba автоматов S_1 и S_2.

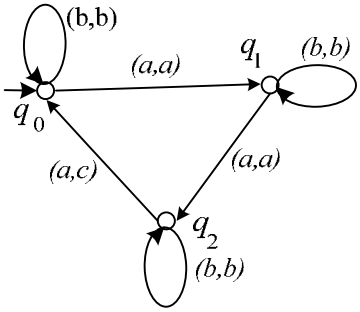


Рис. 25. Граф переходов автомата S_8

2. Определить автоматное отображение для автомата S_8.

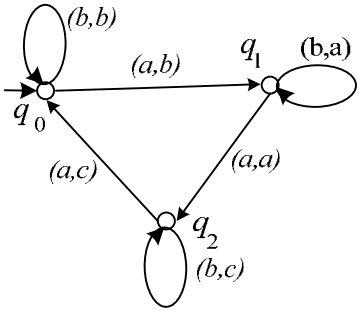


Рис. 26. Граф переходов автомата S_9

3. Автомат Мили S_9 получен из автомата Мура. По первой или второй схеме рассматривался автомат Мура?

4. Минимизировать автомат, заданный совмещённой таблицей переходов и выходов.

	q_0		q_1		q_2		q_3		q_4		q_5	
0	q_2	a	q_5	a	q_5	b	q_4	a	q_3	a	q_2	b
1	q_4	a	q_3	a	q_4	a	q_0	a	q_1	a	q_3	a

5. Пусть для минимизированного автомата Мили $|Q|=n$, $|V_a|=k$, $|V_b|=p$. Оценить число состояний эквивалентного автомата Мура, построенного по второй схеме.

6. Пусть для автомата Мура, построенного по некоторому автомату Мили $|Q|=n$, $|V_a|=k$, $|V_b|=p$. Оценить число состояний исходного автомата Мили.

ГЛАВА 3. СИНТЕЗ АВТОМАТОВ

В разделе рассматриваются различные способы соединения автоматов, понятие правильно построенных логических сетей и способ их построения.

Последовательные автоматные вычисления

Соединяя различные автоматы, работающие последовательно, можно получать автоматные блок-схемы, например схему, представленную на рис. 27. Здесь имеется в виду, что сначала работает автомат S_1, после окончания работы автомата S_2 работает автомат S_3, по результатам работы которого производится возврат к работе автомата S_1 или S_2. Следует отметить, что автоматы в такой схеме должны уметь останавливаться (работа схемы заканчивается, когда заканчивает работу любой из автоматов).

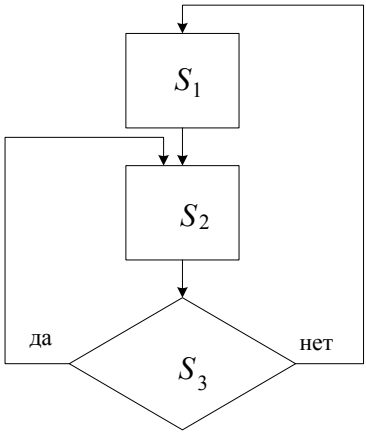


Рис. 27. Автоматная схема для последовательных вычислений

Возможны различные случаи рассмотрения последовательных автоматных схем [2].

1. Алфавиты автоматов не пересекаются. В этом случае действуют обычные правила минимизации для частичных автоматов, и для правильных последовательностей входных символов суммарное число состояний получаемого автомата – $\mathbf{max}\ |\boldsymbol{Q_i}|$. Такое число состояний построенного автомата обусловлено тем, что при не пересекающихся входных алфавитах, любую пару состояний двух разных автоматов $\boldsymbol{A_1}$ и $\boldsymbol{A_2}$ можно объединить. Это объединённое состояние работает как состояние первого автомата, если входной символ принадлежит входному алфавиту первого автомата, и как состояние второго автомата, если входной символ принадлежит входному алфавиту второго автомата.

2. Входные алфавиты исходных автоматов совпадают или включают друг друга. Тогда множество состояний является объединением множества состояний исходных состояний, затем могут производиться эквивалентные преобразования автомата, в этом случае число состояний $\leq \boldsymbol{\Sigma}\ |\boldsymbol{Q_i}|$.

В любом случае блок-схема автоматов, работающих последовательно, – конечный автомат S, значит, множество автоматов замкнуто относительно операции условного и безусловного перехода (следовательно, каждая компьютерная программа и каждый реализуемый программой алгоритм являются автоматом, число состояний которого, однако, может быть очень велико).

Синхронные сети автоматов

Автомат (Мили) называется комбинационным, если для любого входного символа a и любых состояний q_i и q_j выполнено условие $g(q_i, a) = g(q_j, a)$. Неформально, в комбинационном автомате выходной символ не зависит от текущего состояния и определяется входным символом (все состояния автомата в данном случае эквивалентны, т.е. минимизированный автомат будет иметь всего одно состояние).

Комбинационный автомат называется логическим, если его входной алфавит состоит из 2^m двоичных последовательностей длиной m, а выходной алфавит – 2^n двоичных последовательностей длиной n. Тогда функция выхода будет определяться n логическими функциями от m переменных (каждая из функций будет определять один символ в выходной последовательности). Пример построения таких логических функций приведён в заключительном подразделе этой главы.

Поскольку автомат – устройство с входом и выходом, то присоединение к входам одного автомата выходов другого автомата образует сеть, или схему автоматов.

Под состоянием сети из m автоматов $S_1, S_2, \ldots, S_m$ понимается вектор $(q_i^1, \ldots, q_i^m)$, где каждое q_i^j – состояние автомата j в момент времени i. В общем случае число состояний автомата, полученного в результате построения сети, равно произведению чисел состояний исходных автоматов.

Определим, является ли полученная схема автоматом.

Рассмотрим отображение времени в такой схеме. Один из способов введения времени – синхронный: определяются такты работы автоматов, границы тактов нумеруются натуральными числами, начиная с 0. Длина такта рассматривается как единица времени. В этом случае входное слово является временной последовательностью символов (импульсов). Интервал между соседними импульсами – длина такта. Слово длины k будет обрабатываться за k тактов. Входная информация (последовательность входных символов) представляется как $a(t)$, где t – номер такта.

Автоматная функция f реализуется с задержкой; $f(q(t),a(t))=q(t+1)$, $q(0)$ может быть задано отдельно (вместо задания q_0 в инициальных автоматах), обычно $g(q(t), a(t))=b(t)$ (иногда $g(q(t), a(t))=b(t+1)$, тогда задаётся $b(0)$).

Рассмотрим различные виды соединения автоматов.

Пусть даны автоматы $S_1 =<A_1, Q_1, V_1, q_0^1, F_1, G_1>$ и $S_2 =<A_2, Q_2, V_2, q_0^2, F_2, G_2>$. Рассмотрим различные типы соединений автоматов.

Параллельное соединение автоматов

1. С разделительными входами и алфавитами A_1 и A_2 (рис. 28, а).

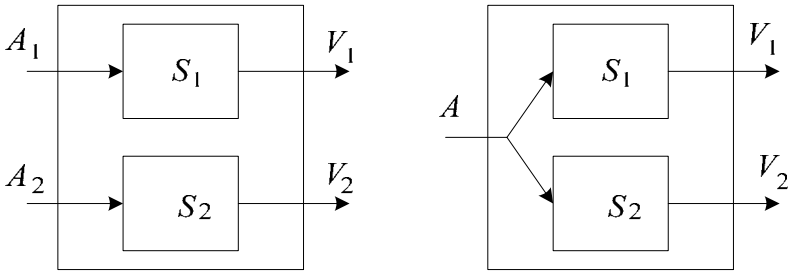
В получаемом автомате $S =<A, Q, V, q_0, F, G>$ входной алфавит $A= A_1 \times A_2$, внутренний алфавит $Q = Q_1 \times Q_2$, выходной алфавит $V=V_1 \times V_2$, $q_0 = (q_0^1, q_0^2)$. S называется прямым произведением автоматов S_1 и S_2. В этом случае $a=(a^1, a^2)$; здесь верхний индекс означает отнесение к соответствующему алфавиту. Функция переходов $f((q^1, q^2), (a^1, a^2)) = (f_1(q^1, a^1), f_2(q^2, a^2))$.

Очевидно определение функции выходов при таком соединении автоматов: $g((q^1, q^2), (a^1, a^2)) = (g_1(q^1, a^1), g_2(q^2, a^2))$.

Мы рассмотрели случай двух входов и двух выходов. Может быть рассмотрен случай произвольного числа входов и выходов.

2. С общим входом и алфавитом A (рис. 28,б).

В получаемом автомате $S = <A, Q, V, q_0, F, G>$ функция переходов $f((q^1, q^2), a) = (f_1(q^1, a), f_2(q^2, a))$. Определение выходов и в данном случае очевидно: $g((q^1, q^2), a) = (g_1(q^1, a), g_2(q^2, a))$.



а) б)

Рис. 28. Параллельное соединение автоматов : а – с разделительными входами, б – с общим входом

Последовательное соединение автоматов

Условное изображение последовательного соединения автоматов S_1 и S_2 представлено на рис. 29.

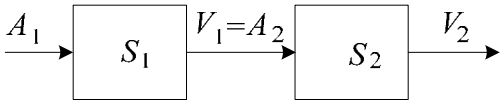


Рис.29. Последовательное соединение автоматов

В получаемом автомате $S = <A, Q, V, q_0, F, G>$ $A=A_1$, $V=V_2$, $V_1=A_2$, $Q= Q_1 \times Q_2$. Для F и G существенна задержка g_1. Если задержка g_1 равна 0, т.е. $g_1(q^1(t), a(t)) = v^1(t)$, то $q(t+1) = (q^1(t+1), q^2(t+1)) = (f_1(q^1(t), a(t)), f_2(q^2(t), g_1(q^1(t), a(t))))$. Таким образом, зависимость $q(t+1)$ существует только от $q(t)$, $a(t)$, т.е. состоя-

ние на данном такте определяется только состоянием и входом на предыдущем такте. При этом состояние $q(t+1) = f(q(t), a(t))$, выход $g((q^1,q^2), a)=g_2(q^2, g_1(q^1, a))$.

Если же задержка g_1 равна **1**, т.е. $g_1(q^1(t), a(t))=v^1(t+1)$, то $q(t+1) = (f_1(q^1(t), a(t)), f_2(q^2(t), g_1(q^1(t-1), a(t-1))))$, и такой простой зависимости, как для прошлого случая, нет.

Пример. Рассмотрим схему последовательного соединения (диаграмма автоматов представлена на рис. 30) двух элементов задержки, воспроизводящих вход через один такт. S_1' и S_2' имеют по два состояния, начальное значение выхода равно **0**. Для автомата, получаемого в результате соединения двух элементов задержки автоматное отображение будет $S(\alpha)=00\alpha_{-2}$ (α_{-2} означает, что отбрасываются два последних символа последовательности α).

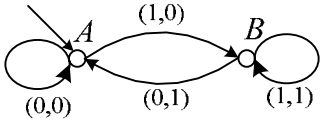


Рис. 30. Граф переходов автомата единичной задержки

Таблица переходов автомата, реализующего единичную задержку:

	q_0	q_1
0	q_0, 0	q_0, 1
1	q_1, 0	q_1, 1

Считаем, что задержка g равна **0**, $g(q(t), a(t))= v(t)= q(t)$.

Обозначим состояние (q_i, q_j) через $i\ j$. Тогда таблица переходов/выходов результирующего автомата следующая:

	00	01	10	11
0	00,0	00,1	01,0	01,1
1	10,0	10,1	11,0	11,1

Граф переходов построенного автомата S_{10} представлен на рис.31.

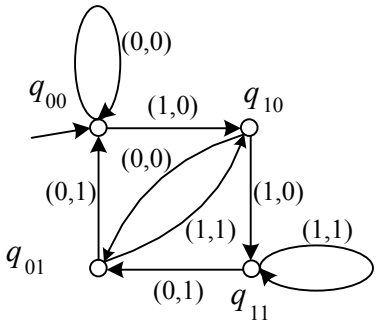


Рис. 31. Граф переходов построенного автомата

Таким образом, цепь из двух элементов задержки описывается автоматом без задержки.

Соединение автоматов с обратной связью

Общая схема представлена на рис. 32. Здесь ***D*** **–** элемент задержки, передающий выходной сигнал на следующем такте.

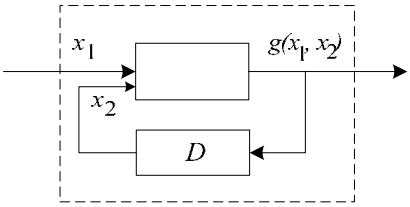


Рис.32. Схема автомата с обратной связью

Если рассматривать вариант обратной связи без задержки, то могут возникать противоречия. Например, если $g(x_1,x_2)$ – функция Шеффера $\overline{x_1} \vee \overline{x_2}$ и $v(t)=0$, тогда $x_2=0$, значит, $v(t)=1$, а при $x_1=1$ это даёт $v(t)=0$. Возникает противоречие в определении состояния автомата, т.е. в реальности состояние будет не определено. Поэтому в общую схему автомата вводится задержка (рис.32).

Представление автомата в виде сети

Всякий автомат при синхронной интерпретации может быть реализован как сеть, составленная из комбинационных автоматов и элементов задержки.

На рис. 33 приведена схема для автомата с функциями

$$\begin{cases} q(t+1) = f(q(t), a(t)); \\ v(t) = g(q(t), a(t)) \end{cases}.$$

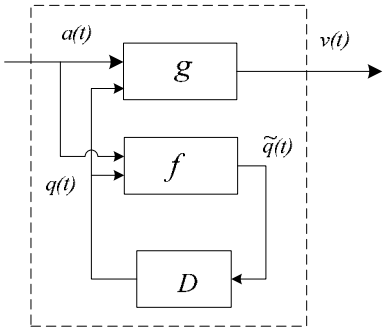


Рис.33. Схема автомата в виде сети

Здесь:

g – комбинационный автомат с входным алфавитом $A \times Q$ и выходным алфавитом V,

f – комбинационный автомат с входным алфавитом $A \times Q$ и выходным алфавитом Q,

D – блок задержки (на один такт).

D реализуется автоматом Мура, входной и выходной алфавит которого $Q = \{q_1, q_2, \ldots, q_n\}$, множество состояний этого автомата $R = \{r_1, r_2, \ldots, r_n\}$, $|R| = |Q|$, функции $g(r_i) = q_i$, $f_D(r_i, q_j) = r_j$.

Частный случай D – двоичный элемент задержки, тогда $q'(t) = f(q(t), a(t)) = q(t+1)$.

В важном частном случае, когда A, V, Q состоят из двоичных наборов, f и g – логические комбинационные автоматы, двоичные выходы которых в момент t являются логическими функциями от двоичных переменных, образующих наборы $a(t)$ и $q(t)$, D – параллельное соединение элементов задержки. Число элементов задержки равно длине вектора Q, а число состояний D равно мощности внутреннего алфавита $|Q| = 2^n$.

Так как произвольные конечные алфавиты могут быть закодированы двоичными наборами, то получается следующая теорема.

Теорема 5. Любой конечный автомат при любом двоичном кодировании может быть реализован синхронной сетью из логических комбинационных автоматов и двоичных задержек, причем число задержек не может быть меньше $\log_2 |Q|$.

Правильно построенные логические сети

Сеть из логических блоков и элементов задержки будем называть правильно построенной логической сетью (ППЛС), если

1) к каждому входу блока сети присоединён не более чем один выход блока сети (однако допускается присоединение выхода более чем к одному входу, т.е. допускается разветвление выходов);

2) в каждом контуре обратной связи, т.е. в каждом цикле, образованном блоками и соединениями между ними, имеется не менее одного элемента задержки.

Входами такой сети называются те входы блоков, к которым не присоединены никакие выходы, выходами сети называются те выходы блоков, которые не присоединены ни к каким входам.

ППЛС, реализующая автоматное отображение, схематически представлена на рис. 34.

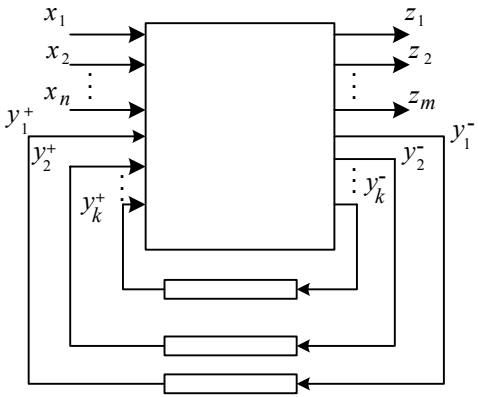


Рис. 34. Схематическое изображение правильно построенной логической сети, реализующей автоматное отображение

На представленной схеме:

$\boldsymbol{M_x}$ – множество входных векторов,

$\boldsymbol{M_z}$ – множество выходных векторов,

$\boldsymbol{M_{y-}}$ – множество векторов, характеризующих входные каналы обратной связи (памяти),

$\boldsymbol{M_{y+}}$ – множество векторов, характеризующих выходные каналы обратной связи (памяти).

Каждый из каналов в случае $\boldsymbol{k}$-значной логики может находиться в одном из $\boldsymbol{k}$ состояний из множества $\{0, 1,\ldots, \boldsymbol{k}\text{-}1\}$. Обычно рассматривается бинарная логика, $\boldsymbol{k}$=2.

Правила вывода, соответствующие автомату, можно определить как подстановку

$$\boldsymbol{XY^+ \rightarrow Z\,Y^-,\ Y^+(t=0)=Y_0^+,\ Y^+(t+\tau)=Y^-(t)}.$$

Состояния каналов обратной связи будем называть внутренними состояниями автомата, а τ – временем перехода из одного состояния в другое, причём τ может быть постоянной для данного автомата или же зависеть от изменения X. В первом случае автомат называется синхронным, во втором – асинхронным.

При заданном значении Y_0^+ последовательность входных векторов X (входная последовательность) однозначно определяет последовательность векторов Z (выходную последовательность).

Например, построим функции переходов и выходов для правильно построенной сети, реализующей автомат S_D' (граф переходов представлен на рис.15). Таблица переходов автомата приведена ниже:

Вход\состояние	K_1	K_3	K_4
0	K_1, 1	K_4, 0	K_1, 0
1	K_3, 0	K_4, 1	K_3,1

Для построения логических функций необходимо ввести обозначения состояний, входов и выходов. Входы и выходы являются бинарными величинами, их можно не переобозначать. Обозначим состояния : K_1– 00, K_3– 10, K_4 – 11. Первый разряд кода состояния представляет собой значение переменной y_1, второй разряд – значение переменной y_2. Входной сигнал кодируется одним символом, это значение переменной x_1, выходной сигнал – так же один символ, это значение переменной z_1.

Тогда таблица переходов может быть развернута во временную зависимость:

x_1	y_1^+, y_2^+	y_1^-, y_2^-	z_1
0	**00**	**00**	**1**
1	**00**	**10**	**0**
0	**10**	**11**	**0**
1	**10**	**11**	**1**
0	**11**	**00**	**0**
1	**11**	**10**	**1**

С использованием этих обозначений можно построить логические функции для функций переходов и выхода правильно построенной логической схемы, реализующий этот автомат, при использовании для функции задержки автомата Мура, работающего по второй схеме, с таблицей:

$a(t)$	$q(t)$	$q(t+1)$	$z(t)$
0	**0**	**0**	**0**
1	**0**	**1**	**0**
0	**1**	**0**	**1**
1	**1**	**1**	**1**

$$y_1^- = x_1 \vee y_1^+ \overline{y_2^+},$$

$$y_2^- = y_1^+ \overline{y_2^+},$$

$$z_1 = x_1 y_1^+ \vee \overline{x_1}\,\overline{y_1^+}\,\overline{y_2^+}.$$

В общем случае вид функций зависит от выбранного способа кодирования состояний, входного и выходного векторов, и строится по таблице логической функции, соответствующей рассматриваемой переменной.

Вопросы и упражнения

1. Чему равно число состояний автомата, полученного параллельным соединением с разделительными входами двух автоматов, каждый из которых имел три состояния?

2. При каких условиях может быть построено параллельное соединение автоматов с общим входом?

3. Как по построенному параллельному соединению автоматов с разделительными входами построить автомат с общим входом (считаем, что необходимые условия для возможности такого построения выполнены)?

4. При каких условиях можно построить последовательное соединение автоматов?

5. При каких условиях для последовательного соединения автоматов можно зависимость состояния в момент времени $t+1$, $q(t+1)$, выразить как $f(q(t), a(t))$?

6. Построить логические функции для реализации правильно построенной логической сетью автомата S_{11}, граф переходов которого представлен на рис.35.

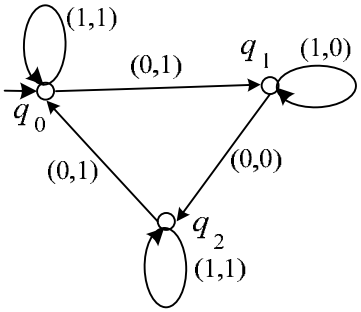


Рис.35. Граф переходов автомата S_{11}

7. Пусть при параллельном соединении автоматов один из автоматов – комбинационный. Как можно оценить число состояний полученного автомата при соединении с общим входом? С разделительными входами?

8. Пусть автомат Мили имеет три состояния, пять входных и два выходных символа. Какова будет размерность векторов при двоичном кодировании состояний, входных и выходных сигналов? Сколько логических функций потребуется, чтобы описать его функции переходов и выходов?

9. Построить последовательное и оба вида параллельного соединения автоматов S_{11} и S_{12}. Граф переходов автомата S_{11} представлен на рис.35. Таблица переходов автомата S_{12} представлена ниже**.**

	q_0	q_1
0	q_0, 0	q_0, 1
1	q_1, 1	q_1, 0

10. Определить число состояний автомата, полученного в результате последовательного соединения двух автоматов, для каждого из которых число состояний равно 4, функция выходов первого автомата – без задержки.

ГЛАВА 4. ЯЗЫКИ И ГРАММАТИКИ

Рассматривается ряд основных понятий математической лингвистики, такие как алфавит, формальный язык и грамматики. Приводятся определения и свойства основных операций над языками. Разобраны различные способы представления языков, особое внимание уделяется регулярным выражениям и основным тождествам, их связывающим. Представлена классификации грамматик.

Алфавит, слова, операции над словами

Пусть $V=\{v_1, v_2,\ldots, v_n\}$, $n\geq 1$ – некоторый алфавит. Тогда любая последовательность $x_1x_2\ldots x_k$, $k\geq 0$, где $x_i\in V$, $1 \leq i \leq k$, слово в алфавите V; при $k=0$ –получается пустое слово, обозначим его λ. Множество всех слов алфавита V обозначается V^*.

Например, пусть $V_1=\{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}$. Тогда V_1^* содержит все числа и пустое слово. Допускаются «числа», которые начинаются с **0** и последовательности нулей.

Слово $X=x_1\ldots x_k$ графически совпадает со словом $Y=y_1\ldots y_m$, если $x_i\in V$ $(1 \leq i \leq k)$, $y_j\in V$ $(1 \leq j \leq m)$, $m=k$, и для любого i ,$1 \leq i \leq k$, $x_i=y_i$. Графическое совпадение слов обозначается $X=Y$.

Длиной слова X (обозначается $|X|$) называется число вхождений символов в слово X. Если $X = x_1 \ldots x_k$, то $|X| = k$. По определению, $|\lambda| = 0$.

Конкатенацией слов $X = x_1 \ldots x_k$ и $Y = y_1 \ldots y_l$ называется слово $Z = XY = x_1 \ldots x_k \; y_1 \ldots y_l$. Например, конкатенацией слов «кон» и «такт» будет слово «контакт».

Если слово $Z = XY$, то X называется началом, а Y – концом слова Z.

Свойства конкатенации:

1. λ является единицей для конкатенации, т.е. для любого слова X верно, что $X\lambda = \lambda X = X$.
2. Операция конкатенации является ассоциативной, т.е. $(XY)Z = X(YZ)$.
3. Операция конкатенации не является коммутативной, $XY \neq YX$.

Для конкатенации, как и для произведения, конкатенация n одинаковых слов X обозначается X^n. Считаем, что $X^0 = \lambda$ для любого слова X. Множество V^* всех слов алфавита V является полугруппой относительно операции конкатенации.

Языки. Операции над языками

Произвольное множество цепочек над алфавитом V, иначе любое подмножество свободной полугруппы V^*, называется формальным языком над V. Поэтому язык может быть задан как любое множество:

перечислением элементов;

ограничивающим свойством;

через известные множества;

порождающей процедурой.

Например, для любого V множество слов четной длины является языком. Множество слов нечетной длины также является языком, но в отличие от первого — не замкнутым относительно операции конкатенации. Будем обозначать языки буквой L (с индексами или без них). Рассмотрим операции над языками.

Так как язык является множеством, то к формальным языкам применимы все операции над множествами, для этих операций выполняются все законы теории множеств.

В частности, для любого языка L справедливо $\varnothing \subseteq L$.

Теоретико-множественные операции

Пусть L_1 и L_2 – два языка над алфавитом V. Язык L называется объединением этих языков (обозначается $L = L_1 \cup L_2$), если $L = \{x \mid x \in L_1 \vee x \in L_2\}$.

Пересечением языков L_1 и L_2 называется язык L (обозначается $L = L_1 \cap L_2$), такой, что $L = \{x \mid x \in L_1 \,\&\, x \in L_2\}$.

Дополнением языка L до V^* называется язык L_1 (обозначается $\overline{L}$), такой, что $L_1 = \{x \mid x \in V^* \,\&\, x \notin L_1\} = V^* \setminus L$.

Специфические операции

Произведением (иначе конкатенацией) языков L_1 и L_2 называется язык L (обозначается $L = L_1 L_2$), $L = \{xy \mid x \in L_1 \,\&\, y \in L_2\}$.

Например, $L_1 = \{ac, a\}$, $L_2 = \{cb, b\}$, тогда $L_1 L_2 = \{acb, accb, ab\}$.

Свойства операции конкатенации:

1) операция умножения языков ассоциативна:
$L_1 (L_2 L_3) = (L_1 L_2) L_3$;

2) операция умножения языков дистрибутивна относительно объединения:
$L (L_1 \cup L_2) = L L_1 \cup L L_2$,
$(L_1 \cup L_2) L = L_1 L \cup L_2 L$;

3) операция умножения языков не коммутативна: $L_1 L_2 \neq L_2 L_1$.

4) $L \{\lambda\} = \{\lambda\} L = L$;

5) $L(L_1 \cap L_2) \subseteq L L_1 \cap L L_2$, в общем случае равенства нет, что легко показать, подобрав соответствующий пример;

6) для любого языка L : $L \varnothing = \varnothing L = \varnothing$;

7) если $L_1 \neq \varnothing$, $\lambda \notin L$, то $L_1 \not\subset L L_1$.

Приведём доказательство последнего утверждения.

1. $L=\varnothing$, тогда $L_1 \not\subset L\ L_1=\varnothing$.

2. $L\neq\varnothing$, тогда язык L содержит некоторые слова. Пусть минимальная длина слова языка L есть m, $m>0$ (т.к. $\lambda\notin L$) . Обозначим минимальную длину слова языка L_1 буквой n ($n\geq 0$). Тогда минимальная длина слова языка $L\ L_1$ равна $m+n$, значит, слова длины n, принадлежащие L_1 (таких слов не менее одного), не содержатся в языке $L\ L_1$, поэтому $L_1 \not\subset L\ L_1$.

В силу ассоциативности операции произведения, как и в случае конкатенации цепочек, $L = \underbrace{L_1\ L_1 \ldots L_1}_{n \text{ раз}}$ записывается как $L=L_1^n$.

По определению, $L^0=\{\lambda\}$.

Отметим, что $\{\lambda\} \neq \varnothing$. так как $\varnothing$ (пустой язык) вообще не содержит никаких цепочек.

Итерацией языка L_1 называется язык L (обозначается $L=L_1^*$), $L=L_1^0 \cup L_1^1 \cup L_1^2 \cup \ldots = \bigcup_{n=0}^{\infty} L_1^n$. Как нетрудно видеть, итерация алфавита V^* (алфавит можно рассматривать как язык, состоящий из односимвольных слов) образует язык, состоящий из всех возможных цепочек, составленных из символов алфавита V. Это обстоятельство и объясняет, почему V^* используется для обозначения множества всех слов над алфавитом V.

Вводится также операция усеченной итерации. L называется усеченной итерацией языка L_1 (обозначается $L=L_1^+$), если

$$L= L_1^1 \cup L_1^2 \cup \ldots = \bigcup_{n=1}^{\infty} L_1^n \ .$$

Регулярные множества и регулярные выражения

Определим некоторый класс языков — регулярные множества. Этот класс часто используется для описания синтаксиса некоторых конструкций языков программирования.

Пусть V_T – конечный алфавит. Регулярные множества в алфавите V_T определяются рекурсивно:

1) $\varnothing$ – регулярное множество;

2) $\{\lambda\}$ – регулярное множество;

3) $\{a\}$ – регулярное множество для любого $a \in V_T$;

4) если P и Q – регулярные множества, то регулярными множествами являются и $P \cup Q$, PQ, P^*;

5) никаких других регулярных множеств нет.

Неформально, регулярное множество для заданного алфавита получается из $\varnothing$, $\{\lambda\}$, и букв этого алфавита с помощью конечного числа применений операций "$\cup$", "$\cdot$" и "*" (объединения, конкатенации и итерации, соответственно).

Определим теперь специальную нотацию для задания регулярных множеств.

Регулярные выражения в алфавите V_T, как и регулярные множества, которые они обозначают, определяются рекурсивно:

1) **0** – регулярное выражение, обозначающее регулярное множество $\varnothing$;

2) **1** – регулярное выражение, обозначающее регулярное множество $\{\lambda\}$;

3) если $a \in V_T$, то a – регулярное выражение, обозначающее регулярное множество $\{a\}$;

4) если p и q – регулярные выражения, обозначающие регулярные множества P и Q, то:

а) $(p+q)$ – регулярное выражение, обозначающее регулярное множество $P \cup Q$,

б) (pq) – регулярное выражение, обозначающее регулярное множество PQ,

в) (p^*) – регулярное выражение, обозначающее регулярное множество P^*;

5) никаких других регулярных выражений нет.

Как обычно, когда можно опустить лишние скобки без потери однозначности понимания, мы будем это делать. Так, $a+bba^*$ обозначает $(a+((bb)(a^*)))$. Считается, что *, как и степень в алгебре,

обладает наивысшим приоритетом, операция конкатенации обладает меньшим приоритетом, и наименьший приоритет у операции +.

Очевидно, что для каждого регулярного множества можно построить регулярное выражение, его обозначающее, и наоборот. К сожалению, как будет показано дальше, одному и тому же регулярному множеству может соответствовать бесконечно много регулярных выражений.

Считаем, что регулярные выражения равны (обозначается значком =), если они обозначают одно и то же регулярное множество.

Запишем основные алгебраические тождества для регулярных выражений. Часть из них уже известны, остальные легко доказываются. Если α, β, γ – регулярные выражения, то:

1. $\alpha+\alpha=\alpha$;
2. $\alpha+\beta=\beta+\alpha$;
3. $\alpha(\beta\gamma)=(\alpha\beta)\gamma$;
4. $\alpha+(\beta+\gamma)=(\alpha+\beta)+\gamma$;
5. $\alpha 1=1\alpha=\alpha$;
6. $\alpha 0=0\alpha=0$;
7. $(\alpha+\beta)\gamma=\alpha\gamma+\beta\gamma$;
8. $\alpha(\beta+\gamma)=\alpha\beta+\alpha\gamma$;
9. $\alpha^*=\alpha+\alpha^*$;
10. $(\alpha^*)^*=\alpha^*$;
11. $\alpha^*\alpha^* = \alpha^*$;
12. $\alpha^*\alpha =\alpha\alpha^*$;
13. $\alpha+0=\alpha$;
14. $1^*=1$;
15. $0^*=1$;
16. $(\alpha\beta)^*\alpha=\alpha(\beta\alpha)^*$;
17. $(\alpha^*\beta)^*\alpha^*=(\alpha+\beta)^*=(\alpha^* +\beta^*)^*$;
18. $(\alpha^*\beta)^*=(\alpha+\beta)^*\beta+1$;
19. $(\alpha\beta^*)^*=\alpha(\alpha+\beta)^* +1$;
20. Если $\alpha = \beta^*\gamma$, то $\alpha=\beta\alpha+\gamma$;
21. Если $1\not\subset\beta$ и $\alpha=\beta\alpha+\gamma$, то $\alpha=\beta^*\gamma$.

Последнее тождество является основным при решении уравнений. Его доказательство несложно получить из тождества 20 и 7-го свойства операций над языками.

Задание языков системами уравнений

Формальные языки могут быть заданы также с помощью систем уравнений разного вида.

Систему уравнений с регулярными коэффициентами назовем стандартной над множеством неизвестных $\Delta=\{X_1, X_2,..., X_n\}$, если она имеет вид:

$$\begin{cases} X_1 = \alpha_{10} + \alpha_{11} X_1 + \alpha_{12} X_2 + ... + \alpha_{1n} X_n; \\ X_2 = \alpha_{20} + \alpha_{21} X_1 + \alpha_{22} X_2 + ... + \alpha_{2n} X_n; \\ \dots \\ X_n = \alpha_{n0} + \alpha_{n1} X_1 + \alpha_{n2} X_2 + ... + \alpha_{nn} X_n, \end{cases}$$

где все α_{ij}, $i \in [1,n]$, $j \in [0, n]$ – регулярные выражения. Если какое-либо i-е уравнение не содержит переменную X_j, то достаточно положить соответствующий коэффициент $\alpha_{ij} = \mathbf{0}$, если $\alpha_{ij} = \mathbf{1}$, то этот коэффициент можно не писать.

В общем случае система уравнений не обязательно является стандартной, она имеет вид:

$$\begin{cases} X_1 = f_1(X_1, X_2, \dots, X_n), \\ X_2 = f_2(X_1, X_2, \dots, X_n), \\ \dots. \\ X_n = f_n(X_1, X_2, \dots, X_n), \end{cases}$$

где f_i – конечная функция, X_j –некоторое множество строк над V_T, на множестве $X_1, X_2, \dots, X_n$ определены операции объединения и конкатенации (при этом каждая конкатенация в общей сумме любого уравнения содержит не более одной переменной). Обозначим $X_1, X_2, \dots, X_n$ как $\overline{X}$, а систему $\overline{X} = F(\overline{X})$. Решение системы $\overline{S} = (S_1, S_2, \dots, S_n)$ – совокупность подмножеств V_T^*, такая, что $\overline{S} = F(\overline{S})$.

Система уравнений может иметь бесконечно много решений, определяется минимальное по мощности решение.

Определим $\overline{S} \subseteq \overline{T} =_{Df} \boldsymbol{S}_1 \subseteq \boldsymbol{T}_1$, $\boldsymbol{S}_2 \subseteq \boldsymbol{T}_2$, …, $\boldsymbol{S}_n \subseteq \boldsymbol{T}_n$.

Функция $\boldsymbol{F}$**:** $\boldsymbol{P}(\boldsymbol{A}) \times \mathbf{P}(\boldsymbol{A}) \rightarrow \mathbf{P}(\boldsymbol{A})$ (здесь $\mathbf{P}(\boldsymbol{A})$ **–** множество всех слов в алфавите А) называется монотонно возрастающей, если из $\boldsymbol{A}_1 \subseteq \boldsymbol{B}_1$ и $\boldsymbol{A}_2 \subseteq \boldsymbol{B}_2$ следует, что $\boldsymbol{F}(\boldsymbol{A}_1, \boldsymbol{A}_2) \subseteq \boldsymbol{F}(\boldsymbol{B}_1, \boldsymbol{B}_2)$. Очевидно, что монотонно возрастающими являются функции построения объединения и пересечения множеств, в частности, функция объединения для множеств слов.

Лемма. Операция конкатенации – монотонно возрастающая функция. Доказательство легко построить, основываясь на определении операции конкатенации.

Кроме того, из дистрибутивности операции объединения относительно конкатенации и ассоциативности объединения следует, что $\boldsymbol{F}(\boldsymbol{A} \cup \boldsymbol{B}) = \boldsymbol{F}(\boldsymbol{A}) \cup \boldsymbol{F}(\boldsymbol{B})$ для любой конечной функции, построенной использованием операций конкатенации и объединения.

Теорема 6. Система уравнений $\overline{X} = \boldsymbol{F}(\overline{X})$ имеет решение $\overline{S} = \bigcup_{i=0}^{\infty} \boldsymbol{F}^i(\overline{\varnothing})$. Если $\overline{T}$ – другое решение, то $\overline{S} \subseteq \overline{T}$.

Доказательство.

Так как $\overline{\varnothing} = (\varnothing, \varnothing, \ldots, \varnothing)$, то $\overline{\varnothing} \subseteq \boldsymbol{F}(\overline{\varnothing})$. Легко показать, что если $\overline{A} \subseteq \overline{B}$, то $\boldsymbol{F}(\overline{A}) \subseteq \boldsymbol{F}(\overline{B})$. Поэтому $\boldsymbol{F}(\overline{\varnothing}) \subseteq \boldsymbol{F}(\boldsymbol{F}(\overline{\varnothing}))$ и т.д. Получаем возрастающую последовательность: $\overline{\varnothing} \subseteq \boldsymbol{F}(\overline{\varnothing}) \subseteq \subseteq \boldsymbol{F}^2(\overline{\varnothing}) \subseteq \boldsymbol{F}^3(\overline{\varnothing}) \subseteq \ldots$

Пусть $\overline{S} = \bigcup_{i=0}^{\infty} \boldsymbol{F}^i(\overline{\varnothing})$. Тогда $\overline{S} = \boldsymbol{F}(\overline{S})$. Если $\overline{T}$ – некоторое другое решение, то $\overline{T} = \boldsymbol{F}(\overline{T})$, но $\overline{\varnothing} \subseteq \overline{T}$, значит, $\boldsymbol{F}(\overline{\varnothing}) \subseteq \boldsymbol{F}(\overline{T}) = \overline{T}$. Очевидно, по индукции можно доказать, что $\boldsymbol{F}^i(\overline{\varnothing}) \subseteq \overline{T}$ для всех $\boldsymbol{i}$, следовательно, $\bigcup_{i=0}^{\infty} \boldsymbol{F}^i(\overline{\varnothing}) \subseteq \overline{T}$.

Пример. Рассмотрим систему

$$\begin{cases} X_a = a\,X_a + b\,X_b\,, \\ X_b = b\,X_b + c\,. \end{cases}$$

Для удобства работы обозначим $X_a - x$, $X_b - y$:

$$f_1(x, y) = a\,x + b\,y;$$
$$f_2(x, y) = b\,y + c;$$

$$f_1(\varnothing,\varnothing)=\varnothing; \qquad f_1(\varnothing, c)=bc; \qquad f_1(bc, bc+c)= abc+b(b+1)c$$
$$f_2(\varnothing,\varnothing)=c; \qquad f_2(\varnothing, c)=bc+c; \qquad f_2(bc, bc+c)=(b^2+b+1)c$$

$$f_1(abc+b(b+1)c, (b^2+b+1)c) = (a+1)ab(b+1)c+b(b+1)^2c;$$
$$f_2(abc+b(b+1)c, (b^2+b+1)c) = (b^3+b^2+b+1)c;$$

$$f_1((a+1)ab(b+1)c+b(b+1)^2c,\ (b^3+b^2+b+1)c) = (a+1)^2ab(b+1)^2c + +b(b^3+b^2+b+1)c;$$
$$f_2((a+1)ab(b+1)c+b(b+1)^2c, (b^3+b^2+b+1)c) = (b^4 +b^3+b^2+b+1)c\,.$$

Откуда получаем

$$y=b^*c,$$
$$x=a^*bb^*c\,.$$

Тем не менее, основной способ решения стандартной системы уравнений (в данном случае система является стандартной) – метод последовательного исключения неизвестных, подобный методу Гаусса. Покажем применение метода на том же примере:

$$\begin{cases} X_a = a\,X_a + b\,X_b\,, \\ X_b = b\,X_b + c\,. \end{cases}$$

Из тождества 21 для регулярных выражений получаем

$$\begin{cases} X_b = b^*c\,, \\ X_a = a\,X_a + b\,b^*c = a^*bb^*c\,. \end{cases}$$

Грамматики и их классификация

Грамматики и порождаемые ими языки

Порождающей грамматикой называется упорядоченная четверка объектов $G=< V_N, V_T, S, R>$, где:

V_T – алфавит терминальных или основных символов;

V_N — алфавит нетерминальных или вспомогательных символов ($V_T \cap V_N = \varnothing$);

S – начальный символ или аксиома, $S \in V_N$,

R – конечное множество правил или продукций вида $\varphi \to \psi$, где $\varphi \in V_T^* V_N (V_T \cup V_N)^*$, $\psi \in (V_T \cup V_N)^*$ – различные цепочки, а $\to$ специальный символ, не принадлежащий $V_T \cup V_N$ и служащий для отделения левой части правила φ от правой ψ. Такие символы, которые служат для описания языка, но не принадлежат самому языку, будем называть метасимволами. Т.е. в левой части каждого из правил содержится произвольная цепочка из терминалов и нетерминалов, содержащая хотя бы один нетерминал, значит не может быть пустой. Правая часть состоит из произвольного числа терминалов и нетерминалов и может быть пустой.

Определим, что цепочка ω_1 непосредственно выводима из цепочки ω_0 (обозначается $\omega_0 \Rightarrow \omega_1$), если существуют такие $\xi_1, \xi_2, \varphi, \psi$, что $\omega_0 = \xi_1 \varphi \xi_2$, $\omega_1 = \xi_1 \psi \xi_2$, и существует правило $\varphi \to \psi$ ($\varphi \to \psi \in R$). Иными словами $\omega_0 \Rightarrow \omega_1$, если в ω_0 найдется вхождение левой части какого-либо правила грамматики, а цепочка ω_1 получена заменой этого вхождения на правую часть правила.

Существенно, что при определении отношения непосредственной выводимости, обозначаемого символом $\Rightarrow$ ($\Rightarrow$, разумеется, также метасимвол), не указывается, какое правило нужно применять и к какому именно вхождению цепочки φ (если вхождений этой цепочки несколько). Здесь проявляются характерные черты исчислений, к классу которых относятся и порождающие грамматики. Исчисления представляют собой "разрешения" в отличие от алгоритмов, являющихся "предписаниями".

Определим, что цепочка ω_n выводима из цепочки ω_0 за один или несколько шагов или просто выводима (обозначается $\omega_0 \Rightarrow^+ \omega_n$), если существует последовательность цепочек $\omega_0, \omega_1, \omega_2, \ldots, \omega_n$ ($n>0$), такая, что $\omega_i \Rightarrow \omega_{i+1}$, $i \in \{0, \ldots, n-1\}$. Эта последовательность называется выводом ω_n из ω_0, а n – длиной вывода. Выводимость за n шагов иногда обозначается $\omega_0 \Rightarrow^n \omega_n$. Наконец, если $\omega_0 \Rightarrow^+ \omega_n$ или $\omega_0 = \omega_n$, то пишут $\omega_0 \Rightarrow^* \omega_n$.

Нетрудно видеть, что $\Rightarrow^+$ есть транзитивное, а $\Rightarrow^*$ – транзитивно-рефлексивное замыкание отношения выводимости $\Rightarrow$.

Цепочка ω называется сентенциальной формой, если она совпадает или выводима из начального символа грамматики, т.е. если $S \Rightarrow^* \omega$. Сентенциальная форма, состоящая только из терминальных символов, называется сентенцией.

Множество цепочек в основном алфавите грамматики, выводимых из начального символа, или сентенций грамматики, называется языком, порождаемым грамматикой G, и обозначается $L(G)$:

$$L(G) = \{x \mid S \Rightarrow^* x \;\&\; x \in V_T^*\}.$$

Справедливо, что для любого перечислимого множества слов M существует грамматика G, такая, что $M = L(G)$.

Определим, что грамматики G_1 и G_2 эквивалентны, если $L(G_1)=L(G_2)$.

Например,

$G_1 = <\{S\}, \{a\}, S, \{S \to a, S \to a\ a\ S\}>$, $L(G_1) = \{a^{2n+1}, n \geq 0\}$.

$G_2 = <\{S, A\}, \{0, 1\}, S, R>$, $R = \{S \to 1\ A,\ A \to 1\ A,\ A \to 0\ A,\ A \to 0\}$, $L(G_2)$ – множество четных двоичных чисел, больших нуля.

$G_3 = <\{S, A\}, \{0, 1\}, S, R>$, $R = \{S \to 1\ A\ 0,\ A \to 1\ A,\ A \to 0\ A,\ A \to \lambda\}$, $L(G_2) = L(G_3)$, грамматики G_2 и G_3 эквивалентны.

Для сокращения записи грамматик и выводов будем изображать нетерминальные символы прописными буквами латинского алфавита $A, B, C, \ldots, S$ с индексами или без них, терминальные символы – строчными буквами $a, b, c, \ldots$ и цифрами. Прописными буквами U, V, Z будем обозначать символы, которые могут быть как терминальными, так и нетерминальными; строчными буквами u, v, x, y, z с индексами или без них – цепочки, составленные из терминальных символов, а буквами $\alpha, \beta, \gamma \ldots$ – из любых символов.

Кроме того, для обозначения правил с одинаковыми левыми частями, имеющими вид $\alpha\to\beta_1$, $\alpha\to\beta_2$,…, $\alpha\to\beta_n$, будем пользоваться записью $\alpha\to\beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$.

Правила грамматики вида $\varphi \to \psi$, где $\psi \in V_T^*$, называются заключительными.

Поскольку при использовании принятых обозначений для восстановления полного вида грамматики по списку правил достаточно указать начальный нетерминал, далее грамматики будут задаваться списком правил, первое из которых относится к начальному нетерминалу.

Классификация грамматик

Выделяются определенные классы грамматик. Здесь рассматриваются грамматики класса «0», контекстно-свободные (КС), контекстно-зависимые (КЗ) и автоматные грамматики.

Грамматики класса «0» – грамматики, в которых не накладывается никаких дополнительных ограничений на вид правил.

Контекстно-зависимые (КЗ-грамматики) – грамматики, все правила которых имеют вид $\varphi \to \psi$, где $\varphi \in V_T^* V_N (V_T \cup V_N)^*$, $\psi \in (V_T \cup V_N)^*$ и $|\varphi| \le |\psi|$.

Например, этому классу принадлежит грамматика G_4 с правилами

$$\begin{cases} S \to abA \mid ab, \\ bA \to Ab, \\ aA \to aabA, \\ aA \to aab. \end{cases}$$

Контекстно-свободные (КС-грамматики) – грамматики, все правила которых имеют вид $A \to \psi$, где $A \in V_N$, $\psi \in (V_T \cup V_N)^*$.

Например, это грамматика G_5 с правилами $S \to a\ S\ b \mid ab$, $L(G_5) = L(G_4) = \{a^n b^n, n>0\}$.

Автоматные грамматики (А-грамматики) – это грамматики, все правила которых имеют вид $A \to a$, $A \to a\ B$, $A \to \lambda$, $a \in V_T$, $A, B \in V_N$.

В соответствии с классом грамматики, порождающей язык, классифицируются языки. То есть язык, который может быть порожден КС-грамматикой, называется КС-языком (язык, для которого существует порождающая его КС-грамматика, и не существует А-грамматики, порождающей этот язык).

А-язык – язык, для которого может быть построена порождающая его А-грамматика.

КЗ-язык – язык, который может быть построен только КЗ-грамматикой, и, соответственно, не может быть построен ни КС-, ни А-грамматикой.

Непосредственная классификация грамматик выглядит следующим образом.

Класс А-грамматик включается в класс КС-грамматик. Все грамматики принадлежат классу 0.

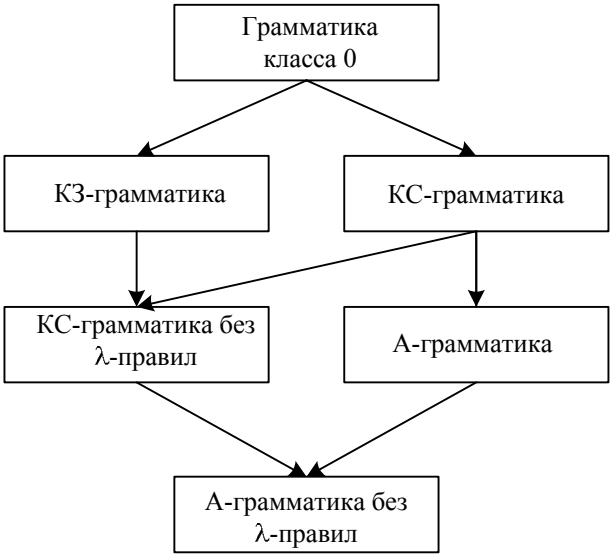


Рис.36. Общий вид классификации грамматик

Однако класс КС-грамматик не включается в класс КЗ-грамматик, так как правила вида $A \to \lambda$ могут принадлежать КС-грамматикам, но не КЗ-грамматикам, так как $|A| > |\lambda|$.

Получаем общий вид классификации грамматик, представленный на рис.36.

Вопросы и упражнения.

1. Что такое слово в алфавите V?

2. Какова длина квадрата слова длиной 4? Длина третьей-степени этого слова?

3. Пусть $X=bbc$, $Y=ac$, $Z=b$. Построить слова XY, $(XY)^2$, X^2Y^2, Y^3, XYZ, XZY, XZ^2Y.

4. Пусть слово X принадлежит языку L_1 и языку L_2. Будет ли слово X принадлежать объединению языков? Пересечению языков?

5. Что такое конкатенация языков? Итерация и усечённая итерация языка?

6. Пусть дан произвольный алфавит V. Верно ли, что $\lambda \in V^*$?

7. Для каких языков $L^*=L^+$?

8. Существует ли язык L такой, что для любого языка L_1 верно, что $L_1 = L\,L_1$?

9. Существует ли язык L такой, что для любого языка L_1 верно, что $L\,L_1 = L$?

10. Даны $L_1=\{ab, ba\}$, $L_2=\{ba, aa, bb\}$. Построить $L_1 \cap L_2$, $L_1 \cup L_2$, $L_1 \setminus L_2$, $L_2 \setminus L_1$.

11. Даны $L_1=\{ab, ba\}$, $L_2=\{bb, aa\}$. Построить $L_1\,L_2$, L_2L_1, L_1^2, L_2^2.

12. Дан $L_1=\{ab, ba\}$. Построить L_1^3, L_1^n, L_1^+, L_1^*.

13. Язык L_3 состоит из различных двоичных чисел. Записать этот язык в виде регулярного выражения.

14. Язык L_4 состоит из нечётных двоичных чисел. Записать этот язык в виде регулярного выражения.

15. Язык L_5 состоит из чётных двоичных чисел. Записать этот язык в виде регулярного выражения.

16. Пусть $|L_1|=n$, $n>0$ $|L_2|=m$, $m>0$. В каком случае $|L_1L_2|=nm$?

17. Пусть $|L_1|=n$, $|L_2|=m$. Оцените величину $|L_1L_2|$.

18. Доказать, что операции объединения и конкатенации являются монотонно возрастающими функциями.

19. Для грамматики G_5 с правилами

$$\begin{cases} S \to aS \mid aB, \\ B \to bB \mid bC, \\ C \to cC \mid c. \end{cases}$$

построить вывод слов abc, a^2b^3c, $a^3b^2c^3$. Записать грамматику полностью. Определить язык, порождаемый грамматикой.

20. Для грамматики G_6 с правилами

$$\begin{cases} S \to abA \mid ab, \\ bA \to Ab, \\ aA \to aabA, \\ aA \to aab. \end{cases}$$

определить язык, порождаемый грамматикой. Определить класс грамматики.

21. Определить класс грамматики G_7 с правилами

$$\begin{cases} S \to aS \mid B, \\ B \to C \mid bC, \\ C \to cC \mid c. \end{cases}$$

и порождаемый ей язык

22. В каком случае грамматики эквивалентны?

23. Могут ли быть эквивалентны КС-грамматика и КЗ-грамматика?

24. Может ли КС-язык быть построен А-грамматикой?

25. Может ли А-язык быть построен КС-грамматикой?

26. В каких случаях А-язык может быть построен КЗ-грамматикой?

ГЛАВА 5. А-ЯЗЫКИ И КОНЕЧНЫЕ ЛИНГВИСТИЧЕСКИЕ АВТОМАТЫ

Рассматриваются различные вопросы, связанные с А-языками и грамматиками. Даются определения диаграммы грамматики, различных типов лингвистических автоматов. Обсуждаются вопросы связи А-грамматик и автоматов, а также соответствия между различными типами автоматов. Приведены теоремы, касающиеся разрешимых свойств А-грамматик.

Диаграмма грамматики

Пусть дана А-грамматика $G=< V_N, V_T, S, R>$. Говорят, что грамматика находится в канонической форме, если все правила грамматики имеют вид $A \to aB$ или $A \to \lambda$. Любая А-грамматика может быть приведена к канонической форме. Грамматика, приведённая к канонической форме, эквивалентна исходной грамматике.

Приведение грамматики к каноническому виду производится следующим образом.

1. Вводим дополнительный нетерминальный символ K, тогда $V_N' = V_N \cup \{K\}$.
2. Заменяем правила вида $A \to a$ на правила $A \to aK$.
3. Вводим дополнительное правило $K \to \lambda$.

Таким образом, все правила грамматики теперь приобрели "стандартный" вид $A \to aB$ или $A \to \lambda$.

Диаграмма А-грамматики – граф с помеченными вершинами и дугами. Множество вершин графа соответствует множеству нетерминалов А-грамматики, приведенной к каноническому виду, а множество дуг – множеству правил грамматики.

Построение диаграммы описывается следующими правилами.

1. Каждому нетерминальному символу поставим в соответствие вершину и пометим ее этим символом.
2. Каждому правилу $A \to aB$ сопоставим дугу из вершины A в вершину B и пометим ее терминальным символом a.

3. Отметим в графе как начальную вершину – вершину, соответствующую начальному символу грамматики, и как заключительные вершины – все такие вершины B, для которых существует правило $B \to \lambda$ (на диаграмме для обозначения заключительных вершин используется символ #).

Пример. Пусть А-грамматика G_8 задана правилами:

$$\begin{cases} S \to aB \mid bC, \\ B \to b \mid bB, \\ C \to aS. \end{cases}$$

Грамматика в канонической форме будет иметь вид:

$$\begin{cases} S \to aB \mid bC, \\ B \to bK \mid bB, \\ C \to aS, \\ K \to \lambda. \end{cases}$$

Диаграмма грамматики приводится на рис.36.

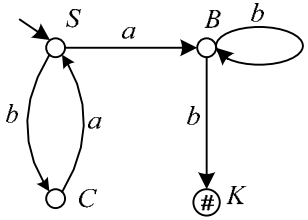


Рис.36. Диаграмма грамматики G_8, приведённой к канонической форме

Очевидно, что существует взаимно однозначное соответствие между грамматикой в каноническом виде и диаграммой.

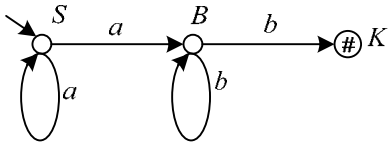


Рис.37. Диаграмма грамматики G_9

Например, рассмотрим диаграмму грамматики, представленную на рис.37. Очевидно, что диаграмме соответствует грамматика G_9 с правилами

$$\begin{cases} S \to aS \mid aB, \\ B \to bK \mid bB, \\ K \to \lambda. \end{cases}$$

Порождение и распознавание цепочек

Лингвистический автомат – это $S_L = \langle Q, V_T, q_0, F, K \rangle$, где

$Q = \{q_0, q_1, \ldots, q_k\}$, $k \geq 0$ – множество состояний автомата (внутренний алфавит),

$V_T = \{a_1, a_2, \ldots, a_m\}$, $m \geq 1$– множество терминальных символов (внешний алфавит) автомата,

q_0 – начальное состояние автомата, $q_0 \in Q$,

$F: Q \times V_T \to Q$ функция переходов,

$K \subseteq Q$ – множество конечных (заключительных) состояний.

Рассмотрим автомат как распознающий, тогда ему соответствует следующая абстрактная модель: входная лента, на которой расположена анализируемая цепочка, считывающая (входная) головка и устройство управления. На каждом шаге обозревается ровно один символ.

Если автомат находится в состоянии q_i, обозревается символ a_j, и $F(q_i, a_j) = q_k$, то считывающая головка перемещается на один символ вправо, автомат переходит в состояние q_k и обозревается следующий символ на ленте. Если же функция $F(q_i, a_j)$ не определена, то входная цепочка не допускается автоматом.

Если в результате прочтения входной цепочки автомат окажется в заключительном состоянии, то считается, что автомат допустил цепочку.

Дадим более строгое определение допускаемой цепочки. Назовём конфигурацией автомата пару $H=(q, x)$, где q — текущее состояние автомата; x — остаток входной цепочки, самый левый символ которой обозревается входной головкой. Определим, что конфигурация (p, x_1) получена из конфигурации (q, x) за один такт (обозначается $(q, x) \vdash (p, x_1)$), если $x= a\, x_1$ и $F(q, a)= p$.

Если $H_0, H_1,\ldots, H_n$ $(n \geq 1)$ – последовательность конфигураций, таких, что $H_i \vdash H_{i+1}$, $i \in \{0,1,\ldots, n-1\}$, то, как и раньше, будем использовать обозначения $H_0 \vdash^{+} H_n$ или $H_0 \vdash^{*} H_n$, если справедливо $H_0 \vdash^{+} H_n \vee H_0=H_n$.

Пусть x — анализируемая цепочка. Начальная конфигурация имеет вид (q_0, x), заключительная – (q_s, λ), $q_s \in K$. Определим, что автомат A допустил цепочку x, если $(q_0, x) \vdash^{*} (q, \lambda)$ и $q \in K$ (использование отношения $\vdash^{*}$ позволяет включить в множество допускаемых цепочек и пустую цепочку λ, если $q_0 \in K$).

Языком $L(A)$, допускаемым конечным автоматом A, называется множество допускаемых им цепочек:

$$L(A) = \{ x \mid (q_0, x) \vdash^{*} (q, \lambda) \ \&\ q \in K\}.$$

Например, для лингвистического автомата $S_A = < \{q_0, q_1\}, \{a, b, c\}, q_0, F, \{q_1\}>$, функция переходов которого

$F(q_0, c)= q_0$,

$F(q_0, a)= q_1$,

$F(q_1, b)= q_1$,

диаграмма представлена на рис. 38.

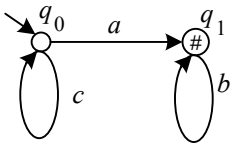


Рис.38. Диаграмма автомата S_A

Язык, распознаваемый автоматом S_A, $L(S_A)= c^* a b^*$.

Цепочка не распознается автоматом, если при распознавании цепочки автомат оказался в таком состоянии, для которого нет перехода по читаемому символу, или в результате прочтения цепочки состояние, в которое перешел автомат – не конечное.

Процесс допускания цепочки автоматом соответствует движению по графу переходов автомата. Цепочка допускается автоматом, если существует путь из начальной вершины в заключительную, при котором последовательно выписанные метки проходимых дуг составляют анализируемую цепочку.

Автоматы S_1 и S_2 эквивалентны, если они распознают один и тот же язык ($L(S_1) = L(S_2)$).

Граф автомата в силу тождественности его структуры с диаграммой грамматик всегда может рассматриваться как диаграмма некоторой грамматики, роль нетерминальных символов в которой будут играть метки состояний автомата. Нетрудно видеть, что грамматика, полученная по графу переходов автомата, при интерпретации последнего как ее диаграммы будет порождать тот же самый язык, который допускается автоматом. В обоих случаях язык однозначно определяется множеством путей из начальной вершины в заключительные, а это множество путей совпадает, так как граф один и тот же. Таким образом, по любому конечному автомату может быть построена эквивалентная **A**-грамматика (т.е. порождающая язык, распознаваемый автоматом), и, следовательно, абстрактно взятый ориентированный граф с помеченными вершинами и дугами, в котором выделена начальная и множество заключительных вершин и удовлетворяются требования однозначности отображения F, может рассматриваться и как диаграмма грамматики, и как граф переходов автомата – все дело в интерпретации.

По графу переходов автомата всегда легко построить эквивалентную грамматику (автомат по грамматике строить сложнее, так как в грамматике одному символу входного алфавита может соответствовать более одного перехода, см., например, рис. 36-37).

Правила грамматики по диаграмме автомата строятся следующим образом:

1. Каждому состоянию автомата сопоставляем нетерминал грамматики.

2. Каждой дуге, соответствующей переходу из состояния P в состояние Q, помеченной терминалом a, сопоставляется правило грамматики $P \to aQ$.

3. Каждому конечному состоянию R сопоставляется правило $R \to \lambda$.

4. Начальному состоянию автомата сопоставляется начальный символ грамматики.

Например, автомату S_A, диаграмма которого представлена на рис.38, соответствует грамматика G_{10} с правилами

$$\begin{cases} S \to cS \mid aA, \\ A \to bA \mid \lambda. \end{cases}$$

где состоянию q_0 сопоставлен нетерминал S, а состоянию q_1 – нетерминал A.

Таким образом, состояния конечного автомата однозначно отображаются в нетерминалы грамматики.

Однако если мы рассмотрим грамматику G_9, диаграмма которой представлена на рис. 36, то однозначность отображения нарушается неоднозначностью переходов, допустимой в грамматиках, но не в автоматах.

Для того чтобы построить соответствующие таким грамматикам автоматы, можно рассматривать функцию переходов F автомата как отображение $Q \times V_T$ в множество подмножеств Q (т.е. в $P(Q)$). При этом $|P(Q)| = 2^{|Q|}$.

Детерминизация недетерминированных автоматов

Будем называть недетерминированным конечным автоматом S пятерку объектов $S = \langle Q, V_T, q_0, F, K \rangle$, где интерпретация Q, V_T, q_0, K такая же, как и раньше, а F – отображение $Q \times V_T$ в $P(Q)$.

Определение цепочек, допускаемых автоматом, остается прежним, но если в детерминированном автомате последовательность конфигураций однозначно определялась заданием входной цепочки, так как из каждой конфигурации автомат мог перейти не более чем в одну конфигурацию, то в недетерминированном случае это

не так. Поэтому при интерпретации определения "цепочка x допущена" как $(q_0,x) \vdash^* (q, \lambda)\&q \in K$, необходимо при анализе цепочки, моделируя работу автомата, перебрать варианты выполнения тактов, чтобы найти вариант (или варианты), который приводит в заключительную конфигурацию. В силу тождественности движений по графу и при порождении цепочки, и при ее допускании, можно утверждать, что для любой грамматики G может быть построен конечный автомат A (в общем случае недетерминированный), такой, что $L(G)=L(A)$. Соответствия между параметрами грамматики и автомата остаются те же.

Определение эквивалентности автоматов является применимым как к детерминированным, так и к недетерминированным автоматам.

Возникает естественный вопрос о соотношении класса языков, допускаемых детерминированными и недетерминированными автоматами. Ясно, что для любого детерминированного автомата A существует недетерминированный A', допускающий тот же самый язык (достаточно в качестве недетерминированного автомата A' взять исходный детерминированный автомат A). Правильность обратного утверждения доказывается в следующей теореме.

Теорема 7. Если $L=L(A)$ для некоторого недетерминированного автомата A, то найдется конечный автомат A', такой, что $L(A')=L(A)$.

Доказательство.

Пусть дан недетерминированный конечный автомат $A = < Q, V_T, q_0, F, K>$. Построим соответствующий детерминированный автомат $A' = <Q', V_T, q_0', F', K'>$

1. $Q' = P(Q)$. При этом множество состояний $\left\{q_{i_1}, q_{i_2}, ..., q_{i_l}\right\}$ будем обозначать как обобщённое состояние $\left[q_{i_1}, q_{i_2}, ..., q_{i_l}\right]$.

2. $q_0' = [q_0]$.

3. $K' = \{\left[q_{i_1}, q_{i_2}, ..., q_{i_l}\right] \mid \left\{q_{i_1}, q_{i_2}, ..., q_{i_l}\right\} \cap K \neq \varnothing\}$.

4. $F'\left(\left[q_{i_1}, q_{i_2}, ..., q_{i_l}\right], a\right) = \bigcup_{j=1}^{l} F\left(q_{i_j}, a\right)$.

Несложно доказать методом математической индукции, что для любого i справедливо:

$([q_0], x\,y) \vdash_{A'}^{i} (B, y) \Leftrightarrow B = \{p \mid (q_0, x\,y) \vdash_{A}^{i} (p, y)\}$ при $|x| = i$.

Значит, для любой цепочки x и для любого $i \in [0,\infty)$

$([q_0], x) \vdash_{A'}^{i} (B, \lambda) \Leftrightarrow B = \{p \mid (q_0, x) \vdash_{A}^{i} (p, \lambda)\}$.

Поэтому, в случае $B \in K'$, т.е. если x – цепочка, допускаемая детерминированным автоматом, то в исходном недетерминированном автомате существует путь из начального состояния в конечное состояние при чтении этой цепочки и, следовательно, $L(A) = L(A')$.

Таким образом, сопоставляя доказанные утверждения, получаем утверждение: Класс А-языков и класс языков, распознаваемых конечными автоматами, совпадает.

Так, например, для недетерминированного автомата, соответствующего грамматике G_9, диаграмма которой представлена на рис.37, детерминированный автомат, порождающий $L(G_9)$, представлен на рис. 39.

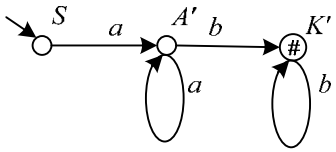
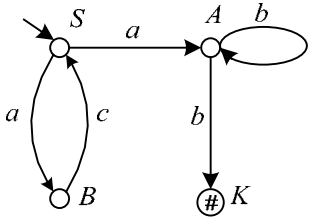


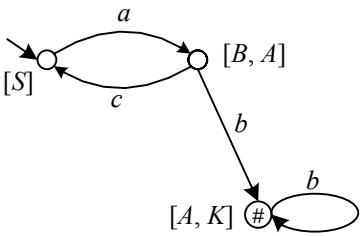
Рис.39. Граф переходов детерминированного автомата, распознающего язык $L(G_9)$

Для этого автомата $F(S, a) = [S, B] = A'$,
$F([S, B], a) = [S, B] = A'$,
$F([S, B], b) = [B, K] = K'$,
$F([B,K], b) = [B, K] = K'$.

Для автомата на рис. 40,а детерминированный автомат представлен на рис.40,б.



а)



б)

Рис.40. Недетерминированный автомат (а) и эквивалентный детерминированный (б)

Алгоритм построения детерминированного автомата по недетерминированному.

1. Строим начальное состояние $q_0' = [q_0]$, помечаем его как начальное.
2. Для каждого состояния q_i', построенного на предыдущем шаге, строим $F(q_i', a)$ для всех $a \in V_T$.
3. Если для какого-нибудь из построенных состояний функция перехода ещё не построена, возвращаемся к шагу 2.
4. Помечаем как конечные все состояния $q_i' = \left[q_{i_1}, q_{i_2}, ..., q_{i_l} \right] \mid \left\{ q_{i_1}, q_{i_2}, ..., q_{i_l} \right\} \cap \mathbf{K} \neq \varnothing$.

Конечность процесса обеспечивается конечностью множества ***P*(*Q*)**.

Автоматы с λ-переходами

Автоматы с λ-переходами естественно возникают в различных приложениях и позволяют представить любой автомат в виде двухполюсников с одним входом и одним выходом, а также строить сети из таких автоматов, сохраняя в них единственный вход и единственный выход. От рассмотренных ранее автоматов они отличаются тем, что в них присутствуют переходы, осуществляемые без чтения входной цепочки (на диаграмме такие переходы обозначаются стрелками, помеченными символом λ, изображение такого перехода представлено рис.41).

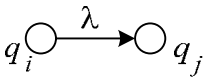


Рис. 41. Пример изображения λ-перехода

Например, рассмотрим автомат с двумя выходами, представленный на рис. 42. Если просто объединить две выходные вершины автомата, то получившийся автомат не будет эквивалентен исходному, так как после построения символа ***a*** в результирующем автомате возможно будет построить символы ***c***, что было невозможно в исходном автомате. Автомат, эквивалентный исходному, и являющийся двухполюсником, представлен на рис. 43. У этого автомата единственное конечное состояние обозначено символом К. Автомат представлен как двухполюсник, так как исходное состояние в автомате всегда единственное, и конечное состояние одно. Иногда в двухполюснике конечные состояния изображаются как  .

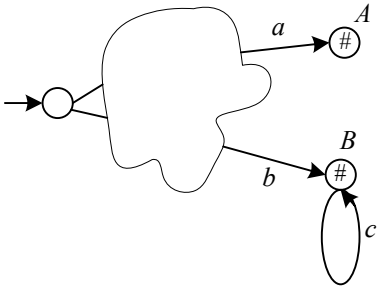


Рис.42. Автомат S_R с двумя конечными состояниями

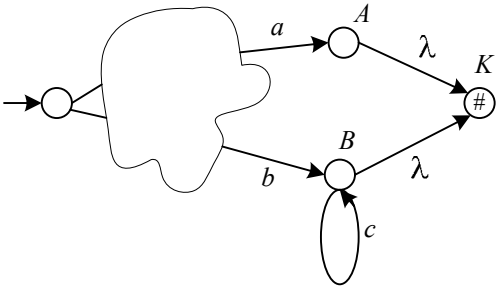


Рис.43. Двухполюсник, соответствующий автомату S_R

Очевидно, что если **L** – А-язык, то ему можно сопоставить некоторый двухполюсник.

Пусть языкам L_1 и L_2 сопоставлены соответствующие двухполюсники (рис.44,а). Тогда их объединению, конкатенации и итерации языка L_1 будут, соответственно, сопоставлены двухполюсники на рис. 44,б, 44,в и 44,г.

Детерминизация автоматов с λ-переходами

Процедура детерминизации автоматов с **λ**-переходами задаётся в следующей теореме.

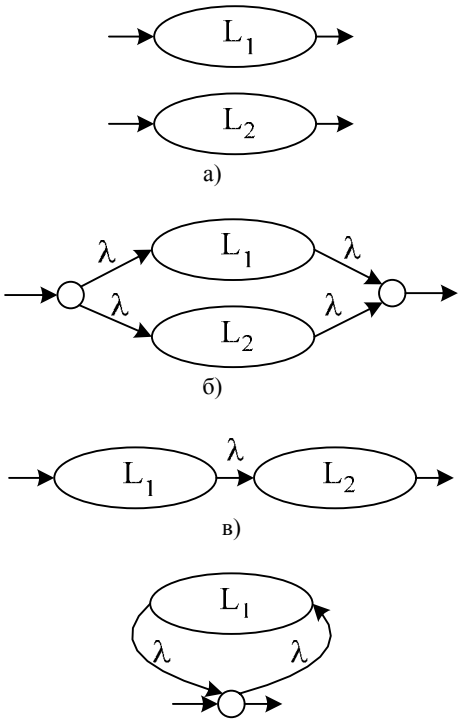


Рис.44. Схематическое представление двухполюсников:
а – для языков L_1 L_2, б – для языка $L_1 \cup L_2$, в – для языка L_1L_2,
г – для языка L^*

Теорема 8. Классы языков, допускаемых детерминированными автоматами и автоматами с **λ**-переходами, совпадают.

Доказательство.

Пусть автомат $A = \langle Q, V_T, q_0, F, K \rangle$ – автомат с **λ**-переходами. Построим соответствующий детерминированный автомат $A' = \langle Q', V_T, q_0', F', K' \rangle$, такой, что $L(A)=L(A')$.

1. $Q' = P(Q)$. При этом множество состояний $\left\{q_{i_1}, q_{i_2}, ..., q_{i_l}\right\}$ будем обозначать как обобщённое состояние $\left[q_{i_1}, q_{i_2}, ..., q_{i_l}\right]$.

2. q_0'=[q_0].

3. $F'(q, a)=\{p \mid (q, ax) \vdash^{+} (p, x)\}$. Построение обобщённых состояний отличается от аналогичного построения для автоматов без λ-переходов тем, что при рассмотрении переходов при прочтении символа a учитываются переходы по λ-дугам, смежным с дугами, помеченными символами a.

4. $K'' = \{ \left[q_{i_1}, q_{i_2}, ..., q_{i_l} \right] / \left\{ q_{i_1}, q_{i_2}, ..., q_{i_l} \right\} \cap \mathbf{K} \neq \varnothing \}$,

$K' = K'' \cup \{q / (q, \lambda) \vdash^{*} (p, \lambda) \& p \in K\}$.

Несложно показать, математической индукцией по числу символов в распознаваемой цепочке, что получаемый таким образом автомат A' переходит при распознавании цепочки **x** в конечное состояние тогда и только тогда, когда существует последовательность переходов в конечное состояние автомата A при распознавании этой же цепочки символов.

Очевидно, что если **L** – А-язык, то ему можно сопоставить некоторый двухполюсник.

Пусть языкам L_1 и L_2 сопоставлены соответствующие двухполюсники (рис.44,а). Тогда их объединению, конкатенации и итерации языка L_1 будут, соответственно, сопоставлены двухполюсники на рис. 44,б, 44,в и 44,г.

Сопоставляя доказанные ранее утверждения, получаем следующую теорему.

Теорема 9. Класс А-языков замкнут относительно операций объединения, конкатенации и итерации.

Соответствие между А-языками и регулярными выражениями

Как соотносятся А-языки и регулярные выражения, определяется в следующей теореме.

Теорема 10 (Клини). Каждому **А**-языку над V соответствует регулярное выражение над V. Каждому регулярному выражению над V соответствует **А**-язык.

Идея доказательства:

$\boldsymbol{L}$ – регулярное множество $\Rightarrow \boldsymbol{L}$ – А-язык.

1. $\varnothing$ – регулярное множество, ему соответствует грамматика с пустым множеством правил;

2. $\{\lambda\}$ – регулярное множество, ему соответствует грамматика с множеством правил $\{ S \to \lambda\}$;

3. $\{a\}, a \in V_T$ – регулярное множество, ему соответствует грамматика с множеством правил $\{S \to a\}$;

4. Если $\boldsymbol{P}, \boldsymbol{Q}$ – регулярные множества, которым сопоставлены **А**-грамматики, то регулярным множествам $P \cup Q$, $P\,Q$, P^* – также соответствуют **А**-грамматики, что легко показать через соединение двухполюсников, порождающих языки, соответствующие $\boldsymbol{P}$ и $\boldsymbol{Q}$.

$\boldsymbol{L}$ – А-язык $\Rightarrow \boldsymbol{L}$ – регулярное множество.

Пусть есть А-грамматика $G = < V_T, V_N, S, R >$, правила для A_i:

$A_i \to \lambda \mid a_1 \mid a_2 \mid \ldots a_k \mid b_1 A_{j1} \mid b_2 A_{j2} \mid \ldots \mid b_m A_{jm}$,

где $a_s, b_q \in V_T$, $A_{js} \in V_N$. Обозначим X_k – язык, порождаемый грамматикой G_k, в которой в качестве начального символа выбран символ A_k. Тогда множество правил для нетерминала A_i соответствует следующему уравнению:

$$X_i = 1 + a_1 + a_2 + \ldots + a_k + b_1 X_{j1} + b_2 X_{j2} + \ldots + b_m X_{jm}.$$

Действительно, если X_i обозначает язык, порождаемый грамматикой G_i, когда A_i – начальный символ, то, так как возможны выводы $A_i \Rightarrow \lambda$, $A_i \Rightarrow a_1$, $A_i \Rightarrow a_2$, $A_i \Rightarrow a_k$, можно написать, что $a_1, a_2, \ldots, a_k \in X_i$ и, следовательно, $X_i = 1 + a_1 + a_2 + \ldots + a_k + \ldots$ С другой стороны, пусть $A_{jk} \Rightarrow^+ x_{jk}$, т.е. $x_{jk} \in X_{jk}$, тогда возможен вывод $A_i \Rightarrow^+ \Rightarrow^+ b_k A_{jk} \Rightarrow^+ b_k x_{jk}$. Следовательно, $b_k x_{jk} \in X_i$, и это верно для любой цепочки $x_{jk} \in X_{jk}$. Поэтому, дополняя предыдущую запись, можем записать уравнение, соответствующее X_i:

$$X_i = 1 + a_1 + a_2 + \ldots + a_k + b_1 X_{j1} + b_2 X_{j2} + \ldots + b_m X_{jm}.$$

Решением системы уравнений будет набор регулярных выражений.

Полное доказательство проводится индукцией по числу правил грамматики.

Как по регулярному выражению построить А-грамматику?

Конкатенация моделируется последовательным соединением двухполюсников, + – параллельным соединением, * – λ- замыканием. Таким образом, последовательно выполняя операции, получим двухполюсники, соответствующие регулярному выражению. Построенные двухполюсники можно упростить, а затем записать соответствующую А-грамматику.

Например, регулярным выражениям **(*a*+*b*)*c*, (*a*+*b*)**c*, (*ab*+*bc*)*(*ab*+*c*)** будут соответствовать диаграммы, представленные на рис. 45.а, б и в соответственно.

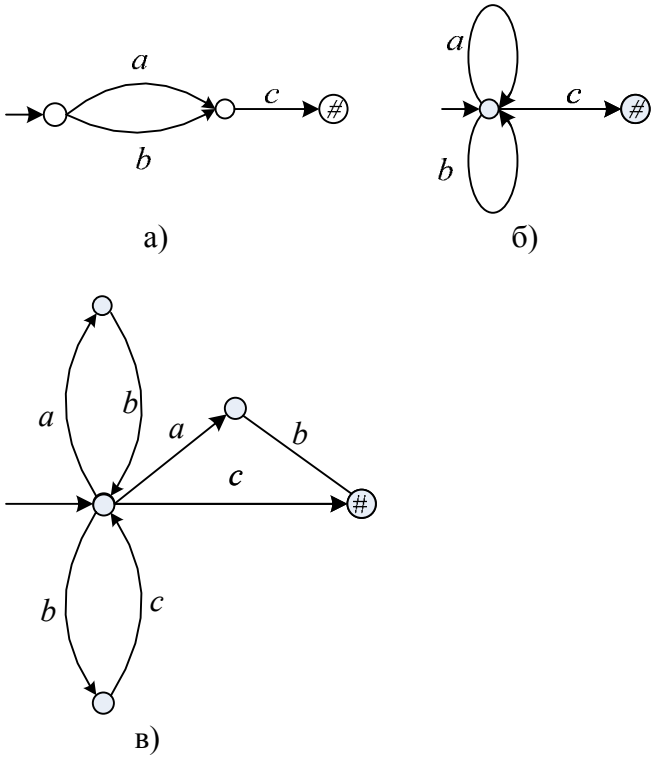


Рис.45. Диаграммы для регулярных выражений: а – **(*a*+*b*)*c*,** б – **(*a*+*b*)**c*,** в – **(*ab*+*bc*)*(*ab*+*c*)**

Обратной задачей является нахождение языка, порождаемого А-грамматикой. Этот язык можно записать в виде регулярного выражения.

Например, имеется А-грамматика G_{11} с правилами:

$$\begin{cases} A \to a\,A \mid bB\,, \\ B \to b\,B \mid c\,. \end{cases}$$

Обозначим язык, порождаемый грамматикой с начальным символом A, – X_a, и язык, порождаемый грамматикой с начальным символом B, – X_b.

Тогда соответствующие уравнения примут вид:

$$\begin{cases} X_a = a\,X_a + b\,X_b\,; \\ X_b = b\,X_b + c\,. \end{cases}$$

Решение этой системы уравнений:

$$\begin{cases} X_a = a^*bb^*c\,, \\ X_b = b^*c. \end{cases}$$

Таким образом, существуют следующие основные способы задания А-языков:

1. А-грамматика.
2. Конечные лингвистические автоматы.
3. Стандартная система уравнений.
4. Регулярное выражение.

Оптимизация автоматов с λ-переходами

Считаем, что автомат задан графом переходов.

1. Если из вершины A исходит единственная дуга и это λ-дуга в вершину B, то вершины A и B можно слить.

2. Если в вершину B входит λ-дуга из вершины A, и это единственная входная дуга вершины B, то вершины A и B можно слить.

3. Вершины, принадлежащие одному компоненту сильной связности в графе, содержащем только λ- дуги, можно слить.

Примеры слияния вершин по правилам 1-2 приведены на рис. 46. Для диаграммы на рис. 46, а, по правилу 1 сливаются C и A, а по правилу 2 – B и A, с получением единственной вершины

[***A,B,C***]. Для диаграммы на рис. 46, в, по правилу 1 сливаются ***C*** и ***A***, а по правилу 2 – ***B*** и ***A***, с получением двух вершин [***A,B,C***] и ***D***.

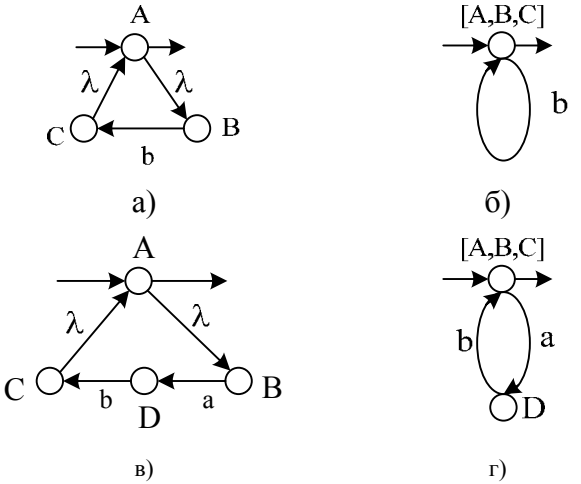


Рис.46 Примеры оптимизации автоматов с λ-переходами

Пример. Пусть автомат S_B , построенный по регулярному выражению **(*ab*+*a**)(*a*+*bc*)**a***, представлен диаграммой на рис. 47, а. Объединим по правилу 1 упрощения автоматов с λ-переходами состояния 4 и 5 и переобозначим состояния, как показано на рис. 47,б.

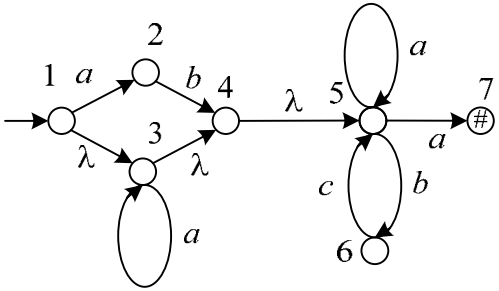
Построим функцию переходов детерминированного автомата S_B':

***F*(*A, a*) = [*B,C, D, F*],** ***F*([*B, C, D, F*], *a*) = [*C, D, F*],**
***F*(*A, b*) = [*E*],** ***F*([*B, C, D, F*], *b*) = [*D, E*],**
***F*(*A, c*) = ∅,** ***F*([*B, C, D, F*], *c*) = ∅,**

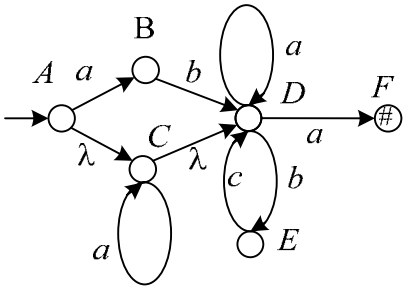
***F*([*E*], *a*) = ∅,** ***F*([*D, E*], *a*) = [*D, F*],** ***F*([*D, F*], *a*) = [*D, F*],**
***F*([*E*], *b*) = ∅,** ***F*([*D, E*], *b*) = [*E*],** ***F*([*D, F*], *b*) = [*E*],**
***F*([*E*], *c*) = [*D*],** ***F*([*D, E*], *c*) = [*D*],** ***F*([*D, F*], *c*) = ∅,**

$F([D], a) = [D, F]$,
$F([D], b) = [E]$,
$F([D], c) = \varnothing$,

$F([C, D, F], a) = [C, D, F]$,
$F([C, D, F], b) = [E]$,
$F([C, D, F], c) = \varnothing$.



а)



б)

Рис.47 Граф автомата, построенного по регулярному выражению**(ab+a*)(a+bc)*a :** а – до оптимизации, б – после оптимизации

Диаграмма детерминированного автомата S_B**'** представлена на рис. 48. Для него **K' = { [*B, C, D, F*] , [*D, F*], [*C, D, F*]}.**

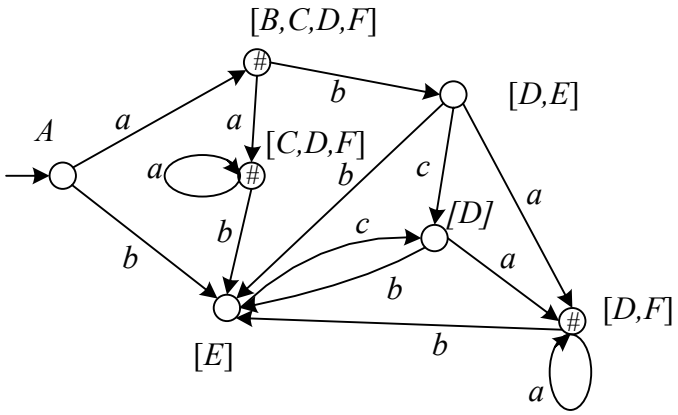


Рис.48. Детерминированный автомат S_B'

Тот же автомат после переобозначения состояний представлен на рис. 49.

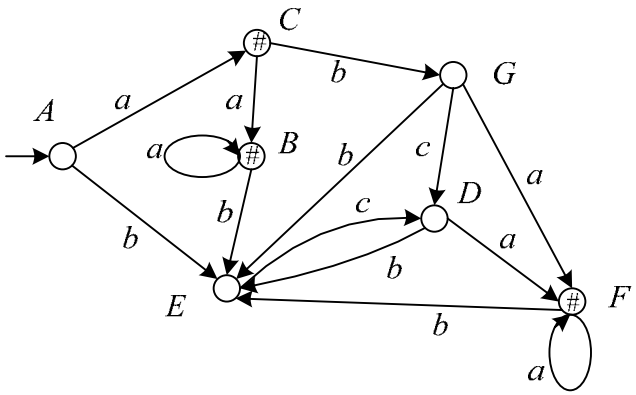


Рис.49. Автомат S_B' после переобозначения состояний

Минимизация числа состояний автомата

Минимизация автомата – переход от автомата к эквивалентному автомату, имеющему минимально возможное число состояний.

Минимизация числа состояний полностью определенных лингвистических автоматов

Минимизация лингвистического автомата может проводиться как методом Хафмена, или же методом таблиц различий. Результаты применений этих методов одинаковы. Мы рассмотрим минимизацию лингвистических автоматов методом Хафмена, метод таблиц различий описан, в частности, в [5].

Алгоритм минимизации лингвистического автомата методом Хафмена.

1. Составляем таблицы переходов и выходов. По строкам указываются входы, по столбцам – состояния, в которые автомат переходит при данном входе.

2. Составляем классы предположительно эквивалентных состояний. В этом случае начальное разбиение происходит на класс конечных состояний и тех состояний, которые не являются конечными.

3. Составляем таблицу переходов, где в ячейках вместо состояний указываются классы, которым принадлежат эти состояния. Затем анализируем полученную таблицу. Если в пределах одного класса состояния ведут себя по-разному, то класс разбивается на соответствующие подклассы, и возвращаемся к шагу 3.

4. Если во всех классах состояния ведут себя одинаково, то это действительно классы эквивалентности, и процедура закончена.

Строится автомат, состояниями которого являются классы эквивалентности исходного автомата.

Пример. Рассмотрим применение метода к лингвистическому автомату $\mathbf{S_C}$, граф переходов которого представлен на рис. 50, имеющему таблицу переходов, представленную ниже, и конечные состояния **C, E**:

	A	*B*	*C*	*D*	*E*
a	***B***	***B***	***A***	***B***	***D***
b	***C***	***D***	***B***	***E***	***B***

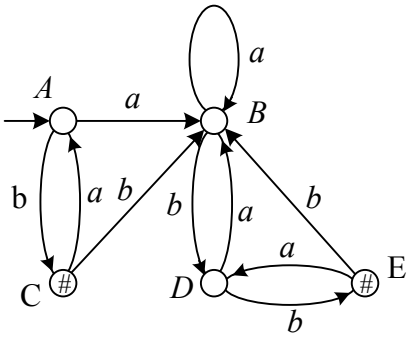


Рис.50. Граф переходов автомата S_C

На первом шаге получаются классы $K_1 = \{C, E\}$ (конечные состояния), и $K_2 = \{A, B, D\}$. Таблица переходов с указанием классов имеет вид (для наглядности в таблицу добавлена строка, с указанием номера класса, которому принадлежит состояние):

№ класса	**2**	**2**	**1**	**2**	**1**
Вход\состояние	***A***	***B***	***C***	***D***	***E***
a	K_2	K_2	K_2	K_2	K_2
b	K_1	K_2	K_2	K_1	K_2

Видно, что класс K_2 разбивается на два класса $K_3 = \{A, D\}$, $K_4 = \{B\}$. Получается таблица переходов:

№ класса	**3**	**4**	**1**	**3**	**1**
Вход\состояние	***A***	***B***	***C***	***D***	***E***
a	K_4	K_4	K_3	K_4	K_3
b	K_1	K_3	K_4	K_1	K_4

Состояния в классах оказываются эквивалентными.

Таблица переходов минимизированного автомата (начальное состояние автомата – $\mathbf{K_3}$):

Вход\состояние	K_3	K_4	K_1
a	K_4	K_4	K_3
b	K_1	K_3	K_4

Граф переходов полученного автомата представлен на рис. 51.

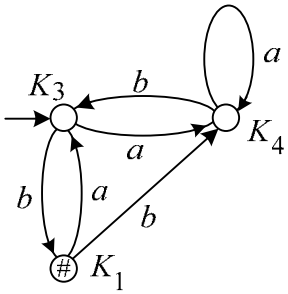


Рис. 51. Граф переходов минимизированного автомата S_C

Минимизация не полностью определённых автоматов

Минимизация не полностью определённых автоматов может проводиться по методу Хафмена. Шаги алгоритма в этом случае такие же, как и для полностью определённых автоматов. Прочерк считаем отдельной буквой.

Процесс минимизации не полностью определённых автоматов рассмотрим на примере автомата $\mathbf{S_B}$, граф переходов которого приведён на рис. 49. При минимизации не полностью определенных лингвистических автоматов начальное разбиение множества состояний делается так же, как для полностью определённого автомата: на конечные и неконечные состояния. Для данного случая получаются классы K_1={***B, C, F***} – конечные состояния, K_2= {***A, D,***

E, G} – неконечные. Далее приводятся таблицы переходов состояний для этого автомата.

Исходная таблица:

	A	***B***	***C***	***D***	***E***	***F***	***G***
a	***C***	***B***	***B***	***F***	-	***F***	***F***
b	***E***	***E***	***G***	***E***	-	***E***	***E***
c	-	-	-	-	***D***	-	***D***

Следующая таблица получена подстановкой классов вместо состояний в таблицу переходов (в дополнительной верхней строке указан номер класса состояния):

№	2	1	1	2	2	1	2
	A	***B***	***C***	***D***	***E***	***F***	***G***
a	K_1	K_1	K_1	K_1	-	K_1	K_1
b	K_2	K_2	K_2	K_2	-	K_2	K_2
c	-	-	-	-	K_2	-	K_2

Далее классы разбиваются на подклассы в случае различия столбцов. Класс $\mathbf{K_2}$ разбивается на 3 класса: $\mathbf{K_3 = \{A, D\}}$, $\mathbf{K_4 = \{E\}}$, $\mathbf{K_5 = \{G\}}$. Получившиеся новые классы подставляются в таблицу переходов:

№	3	1	1	3	4	1	5
	A	***B***	***C***	***D***	***E***	***F***	***G***
a	K_1	K_1	K_1	K_1	-	K_1	K_1
b	K_4	K_4	K_5	K_4	-	K_4	K_4
c	-	-	-	-	K_3	-	K_3

Полученная таблица показывает, что класс K_1 также разбивается на два класса: $K_6 = \{B, F\}$, $K_7 = \{C\}$:

№	3	6	7	3	4	6	5
	A	***B***	***C***	***D***	***E***	***F***	***G***
a	K_7	K_6	K_6	K_6	-	K_6	K_6
b	K_4	K_4	K_5	K_4	-	K_4	K_4
c	-	-	-	-	K_3	-	K_3

И, наконец, разбивается класс K_3 на классы: K_8= {*A*}, K_9={*D*}:

№	8	6	7	9	4	6	5
	A	***B***	***C***	***D***	***E***	***F***	***G***
a	K_7	K_6	K_6	K_6	-	K_6	K_6
b	K_4	K_4	K_5	K_4	-	K_4	K_4
c	-	-	-	-	K_9	-	K_9

Это разбиение не приводит в дальнейшему разбиению классов (для единственного класса K_6, содержащего более одного состояния, столбцы таблицы одинаковы), и получившиеся классы состояний автомата являются классами эквивалентности, а автомат, состояниями которого являются классы эквивалентности исходного автомата, – минимальный. Диаграмма минимизированного автомата S_B', на которой все классы, кроме 6-го, поименованы единственным входящим в них состоянием, а класс K_6 – символом *B*, представлена на рис. 52.

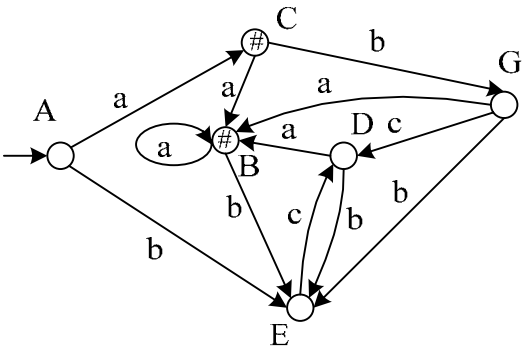


Рис. 52. Результат минимизации автомата S_B

Разрешимые проблемы для А-грамматик

Теорема 11 .Если L – А-язык, и $Z \in L$ имеет длину, не меньшую числа состояний детерминированного автомата, то $Z=WXY$, где $|X| \geq 1$ и $W X^i Y \in L$ для всех $i \geq 0$.

Доказательство.

Пусть есть некоторый А-язык и автомат M_L, порождающий этот язык. $Q=\{ q_0, q_1, q_2, \ldots, q_{n-1}\}$ – множество состояний автомата M_L , n – число состояний автомата, q_0 – начальное состояние автомата.

$F(q_0, Z) \in K$ т.к. $Z \in \mathbf{L}$. $|Z| \geq n$, максимальная длина пути в графе с n вершинами равна $n-1$, а при распознавании цепочки при чтении каждого символа производится переход по одной дуге, следовательно, при распознавании цепочки длиной n не менее чем через одну из вершин путь пройдёт дважды. Поэтому $\exists q_j \in Q$, которое проходится неоднократно при распознании цепочки Z. Пусть X - цепочка, прочитываемая при проходе от q_j до q_j, $|X| \geq 1$ Это, в свою очередь, означает, что $Z=WXY$, $|X| \geq 1$, $F(q_0, W)= q_j$ (W – цепочка, прочитываемая при проходе от начальной вершины до q_j), $F(q_j, X)= q_j$ ($\mathbf{X}$ – цепочка, прочитываемая при проходе от q_j до q_j), $F(q_j, Y) \in K$ (Y – цепочка, прочитываемая при проходе от q_j до некоторой конечной вершины). Но тогда $F(q_0, W X^i Y) \in K$ для всех $i \geq 0$, что требовалось доказать.

Теорема 12. Проблема непустоты для А-грамматик разрешима, т.е., если задана А-грамматика G, то существует алгоритм, позволяющий ответить на вопрос: $L(G)= \varnothing$?

Доказательство.

Например, по предыдущей теореме, если $L(G) \neq \varnothing$, то существует цепочка длины меньшей n, где n – число состояний детерминированного автомата, и перебирая все цепочки длины, не превосходящей n, можно определить, принадлежит ли какая-нибудь из них языку, порождаемому автоматом (хотя это и не оптимальное решение).

Теорема 13. Проблема равносильности А-грамматик разрешима. Т.е., если G_1 и G_2 – А-грамматики, то существует алгоритм, позволяющий определить, $L(G_1)=L(G_2)$?

Доказательство.

В случае равенства языков, порождаемых грамматиками, минимизированные детерминированные автоматы, построенные по этим грамматикам, будут совпадать с точностью до обозначений состояний.

Теорема 14. Проблема конечности языка, порождаемого А-грамматикой, разрешима.

Доказательство.

Обозначим множество состояний, достижимых из состояния A_i, как $H(A_i)$. При этом отношение достижимости не рефлексивно, т.е. $A_i \notin H(A_i)$. Тогда, если $\exists\ A_i$, такое, что $A_i \in H(q_0)\ \&\ A_i \in H(A_i)\ \&\ \&\exists A_j \in H(A_i)\ \&\ A_j \in K$, то язык, порождаемый автоматом, бесконечен.

Хотя с помощью автоматов может быть представлен достаточно широкий класс событий, существуют ограничения на вид событий, задаваемых автоматами, одно из них описывается в следующей теореме.

Теорема 15. Существуют события, не представимые в конечных автоматах: никакая непериодическая последовательность не распознаваема в конечных автоматах (например, последовательность **0101101111011110111110**…). При этом периодические последовательности, в которых могут бесконечно повторяться фрагменты исходной последовательности, не должны распознаваться данным автоматом, т.е. автомат должен из всех последовательностей выделять последовательности требуемого вида.

Доказательство.

Пусть непериодическая последовательность $\alpha=a_1a_2\ldots a_j\ldots$ распознаётся автоматом S с n состояниями. Любой начальный отрезок последовательности $a_1a_2\ldots a_j$ также является допустимым, следовательно, $f(q_0, a_1a_2\ldots a_j)= q_k \in Q_1$ (где Q_1 – множество состояний, соответствующих допустимым цепочкам). При этом автомат проходит последовательность состояний из конечного множества Q_1, значит, для достаточно длинных цепочек некоторое состояние

встретится дважды: $q_s=q_{s+p}$. Это означает, что $f(q_s, a_{s+1}\ldots a_{s+p})=q_s$, поэтому периодическая последовательность также будет распознаваться автоматом, и, следовательно, непериодическая последовательность не может распознаваться конечным автоматом вопреки предположению.

Два основных аспекта работы автомата:

1. Автоматы распознают входные слова, т.е. отвечают на вопрос: $\alpha \in M$? (распознаватели).
2. Преобразуют входные слова в выходные (автоматные отображения).

Сравним эти аспекты работы автоматов. С одной стороны, последовательность ответов распознавателя на входные слова a_1, a_1a_2, $a_1a_2a_3$, … образуют выходное слово, которое является автоматным отображением; с другой стороны, выходные буквы можно разбить на два класса C_1 и C_2, и считать, что слово распознаётся, если $g(q_0,\alpha) \in C_1$. Таким образом, эти два представления автоматов являются эквивалентными.

Вопросы и упражнения

1. Построить диаграмму грамматики G_5 с правилами:

$$\begin{cases} S \to aS \mid aB, \\ B \to bB \mid bC, \\ C \to cC \mid c. \end{cases}$$

2. Для автомата, заданного таблицей переходов с конечным состоянием q_1, построить грамматику, порождающую распознаваемый автоматом язык. Определить язык, порождаемый грамматикой.

	q_0	q_1
a	q_0	q_0
b	q_1	q_1

3. Как определить язык, распознаваемый автоматом?

4. Какова связь между языком, порождаемым автоматом, и графом переходов автомата?

5. Построить автомат, распознающий множество слов, заданных регулярным выражением $(a+ab)^*b+a$.

6. Пусть число состояний автомата равно n. Оценить число состояний эквивалентного минимизированного автомата.

7. Всегда ли конкатенация А-языков является А-языком?

8. Всегда ли объединение А-языков является А-языком?

9. Разрешима ли для А-грамматик проблема эквивалентности?

10. Разрешима ли проблема эквивалентности регулярных выражений?

11. Как проводится минимизация не полностью определённого лингвистического автомата?

12. Сколько конечных автоматов может соответствовать одному регулярному выражению?

13. Как проверить эквивалентность языков, распознаваемых автоматами, по соответствующим минимизированным автоматам?

14. Как по заданному автомату построить регулярное выражение, соответствующее распознаваемому им языку?

15. Доказать, что язык $\{a^n b^n, n \geq 0\}$ не распознаётся конечным автоматом.

16. Как построить конечный автомат для объединения языков?

17. Как построить конечный автомат для конкатенации языков?

ГЛАВА 6. ЭЛЕМЕНТЫ ТЕОРИИ КОДИРОВАНИЯ

В разделе рассмотрена общая проблема кодирования сигналов, основные типы кодирования. Приводится критерий однозначности кодирования для алфавитного кодирования и способ построения кодов с минимальной избыточностью. Дано определение расстояния по Хеммингу для бинарных кодов и изложен способ построения самокорректирующихся кодов. Вопросы кодирования состояний автоматов и возникающие при таком кодировании проблемы не входят в круг вопросов, рассматриваемых в данном пособии.

Основные понятия теории

Кодирование позволяет изучение одних объектов сводить к изучению других. Например, десятичное кодирование чисел позволяет удобным образом определить операции над числами, декартова система координат для геометрических объектов даёт возможность построить их аналитическое представление.

Общий круг проблем кодирования хорошо иллюстрируется на примере систем связи. Схематическое представление системы связи приведено на рис. 53.

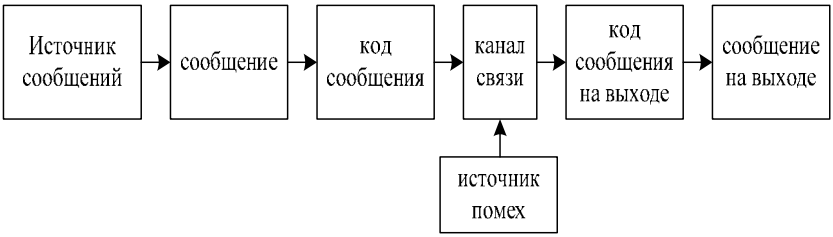


Рис. 53. Общее представление системы связи

Имеется некоторый источник сообщений. Он выдаёт сообщения в некотором алфавите. Для передачи в канале связи это сообщение кодируется с получением кода сообщения. Полученный код сообщения поступает в канал связи. При передаче сообщения по каналу связи в общем случае сообщение может измениться из-за помех. На выходе канала связи получается сообщение, которое затем дешифруется. Рассмотрим подробнее этап кодирования.

Пусть задан алфавит V_a из конечного числа букв. Длина слова A в этом алфавите обозначается как $l(A)$. V_a^+ – множество непустых слов в алфавите V_a. Пусть $S'(V_a) \subseteq V_a^+$. Объект, порождающий слова из S' – источник сообщений, а сами слова – сообщения. Описания источников сообщений могут быть:

- теоретико-множественное, например, множество всех слов заданной длины;

- статистическое, при котором задаются вероятности букв алфавита V_a;

- логическое, т.е. описание S' как некоторого языка.

Пусть задан алфавит $V_b=\{b_1, b_2, \ldots b_n\}$. Пусть B – слово в алфавите V_b, $S'(V_b) \subseteq V_b^+$. Пусть задано отображение $F: S'(V_a) \rightarrow S'(V_b)$. Такое отображение ставит в соответствие произвольному слову A в алфавите V_a слово B в алфавите V_b. Слово B называется кодом сообщения A, а переход от слова A к слову B – кодированием. Отображение F задаётся некоторым алгоритмом. Алгоритм кодирования должен обеспечивать однозначное восстановление исходного сообщения по сообщению на выходе канала связи.

Основными типами кодирования являются следующие.

1. Алфавитное. Это кодирование устанавливает взаимно однозначное соответствие между буквами алфавита V_a и словами алфавита V_b. Это соответствие называют схемой кодирования. Слова, соответствующие буквам алфавита V_a, называются элементарными кодами.

2. Равномерное кодирование. Источник сообщений выдаёт слова одинаковой длины, при этом каждое слово однозначно раскладываются на буквы. Каждому слову ставится в соответствие слово, называемое кодом слова.

Выбор кодов связан с удобством передачи (двоичный удобнее), удобством восприятия, пропускной способностью канала, обеспечением помехоустойчивости кодов, свойствами алгоритмов кодирования.

Код сообщения на выходе в случае тождественного канала (т.е. канала без помех) слово в алфавите V_b. В общем случае на выходе канала связи выполняются функции коррекции ошибок и декодирования.

Рассмотрим алфавитное кодирование для $V_a=\{a_1,a_2,\ldots,a_k\}$. Общий вид схемы кодирования Σ:

a_1 - B_1,

a_2 – B_2,

…

$a_k - B_k$

Здесь B_i $(i \in \{1, 2, \ldots k\})$ – элементарные коды.

Рассмотрим пример алфавитного кодирования с входным алфавитом $\{a_1, a_2\}$, выходным алфавитом $\{b_1, b_2\}$ со схемой кодирования Σ_1:

$a_1 - b_1$,

$a_2 - b_1 b_2$.

Процесс декодирования очевиден.

Критерий однозначности кодирования

Префиксом называется непустое начало слова.

Схема Σ обладает свойством префикса, если для любых i, j $(1 \le i, i \le k, 1 \le j, j \le k, i \ne j)$ элементарный код B_i не является префиксом элементарного кода B_j.

Для схем кодирования, обладающих свойством префикса, может быть построено кодовое дерево. Кодовое дерево – дерево с корнем, ребрам которого приписаны буквы кодового алфавита таким образом, что путь от корня до висячей вершины соответствует элементарному коду. Висячей вершине приписывается соответствующая буква исходного алфавита. Например, пусть дана схема кодирования Σ_2:

$a_1 - aa$,

$a_2 - ab$,

$a_3 - ac$,

$a_4 - ba$,

$a_5 - bbc$.

Эта схема обладает свойством префикса. Кодовое дерево, соответствующее схеме кодирования Σ_2, представлено на рис. 54.

Теорема 16. Если схема кодирования обладает свойством префикса, то алфавитное кодирование будет взаимно однозначным.

Доказательство [1] проводится от противного: если слово имеет две расшифровки, то нарушается свойство префиксности.

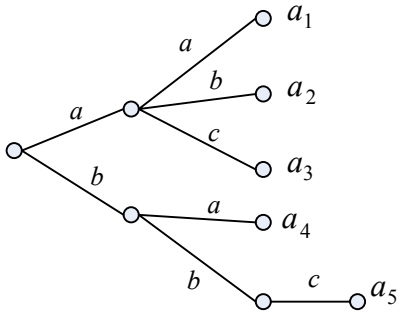


Рис. 54. Кодовое дерево для схемы кодирования $\mathbf{\Sigma_2}$

Можно определить схему, "инверсную" для данной, т.е. код буквы одной схемы получается из кода буквы другой схемы инвертированием (под инвертированием понимаем замену порядка букв в элементарном коде на обратный). Например, для схемы кодирования Σ_1 инверсией является схема $\mathbf{\Sigma_3}$**:**

$\boldsymbol{a_1 - b_1}$**,**

$\boldsymbol{a_2 - b_2 b_1}$**.**

Теорема 17. Если сама схема или её инвертирование обладает свойством префиксности, то алфавитное кодирование будет взаимно однозначным.

Можно придумать такое взаимно однозначное кодирование, при котором ни схема, ни её инверсия не будут обладать свойством префиксности. Например, такова схема кодирования $\mathbf{\Sigma_4}$**:**

$\boldsymbol{a_1 - b_1}$**,**

$\boldsymbol{a_2 - b_1 b_2}$**,**

$\mathbf{a_3 - b_3\ b_1}$.

Тем не менее очевидно, что схема обладает свойством взаимной однозначности. Для декодирования сначала выделяются элементарные коды, соответствующие $\boldsymbol{a_3}$, затем элементарные коды, соответствующие $\boldsymbol{a_2}$, оставшиеся символы $\boldsymbol{b_1}$ соответствуют букве $\boldsymbol{a_1}$. Для правильно построенных кодов процесс дешифровки всегда будет успешен.

Теоремы 16 и 17 дают достаточные условия однозначности декодирования. Однако эти условия не являются необходимыми. Рассмотрим критерий взаимной однозначности кодирования, т.е.необходимые и достаточные условия взаимной однозначности.

Пусть алфавитное кодирование задано схемой **Σ:**

$a_1 - B_1$,

...

$a_n - B_n$.

Для каждого элементарного кода B_i рассмотрим все разложения вида $B_i = \beta' B_{i_1} \ldots B_{i_m} \beta''$ (*), в которых **β', β"** отличны от элементарных кодов. Тривиальные разложения $B_i = \lambda\text{-}B_i\text{-}\lambda$ не рассматриваются.

Строим множество K, содержащее пустое слово и все β, присутствующие в разложениях вида * как в виде префиксов, так и в виде окончаний (т.е. только такие β, которые, во-первых, в каком-нибудь разложении являются префиксом, во-вторых, в каком-нибудь, возможно, том же самом, разложении являются окончанием).

Строится граф **Г(Σ)**. Для каждого нетривиального разложения вида * строится дуга из **β' β"** , помеченная $B_{i_1} \ldots B_{i_m}$.

Пример. Пусть дана схема кодирования Σ_5**:**

1. a_1- *ab*,
2. a_2 – *acb*,
3. a_3- *bc*,
4. a_4 – *abac*,
5. a_5- *babbc*.

Разложения вида * для элементарных кодов:

ab: *a*-λ-*b*,

acb: *a*-λ-*cb*, *ac*-λ-*b*,

bc: *b*-λ-*c*,

abac: λ-*ab*-*ac*, *a*-λ-*bac*,

babbc: *b*-*abbc*-λ, *ba*-λ-*bbc*, *babb*-λ-*c*.

Множество $K(\Sigma_5)=\{\lambda, ac, b\}$. Граф $\Gamma(\Sigma_5)$ представлен на рис.55.

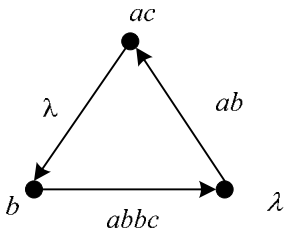


Рис. 55. Граф $\Gamma(\Sigma_5)$

Теорема 18. Алфавитное кодирование со схемой Σ не обладает свойством взаимной однозначности тогда и только тогда, когда $\Gamma(\Sigma)$ содержит ориентированный цикл, проходящий через вершину λ.

На основании этой теоремы можно утверждать, что схема кодирования Σ_5 не обладает свойством взаимной однозначности. В самом деле, цикл, проходящий через вершину λ, даёт неоднозначно декодируемое слово: $abacbabbc = ab\text{-}acb\text{-}ab\text{-}bc = a_1a_2a_1a_3 = abac\text{-}babbc = a_4a_5$.

Другое важное свойство схем кодирования даётся неравенством Макмиллана.

Пусть алфавитное кодирование задано схемой Σ:

$a_1 - B_1$,

...

$a_n - B_n$,

q – число букв в алфавите B, $l_i = l(B_i)$ – длина i-го элементарного кода.

Теорема 19. Если алфавитное кодирование обладает свойством взаимной однозначности, то

$$\sum_{i=1}^{n} \frac{1}{q^{l_i}} \le 1$$ (Неравенство Макмиллана).

Следует обратить внимание, что неравенство Макмиллана дает необходимое, а не достаточное условие взаимной однозначности кодирования. Т.е. если для некоторого кода неравенство Макмил-

лана не выполнено, то код заведомо не является взаимно однозначным. Однако выполнение этого неравенства не говорит о взаимной однозначности кодирования, удовлетворяющего этому неравенству. Так, для схемы Σ_5 q**=3,** l_1**=2,** l_2**=3,** l_3**=2,** l_4**=4,** l_5**=5**, и $\frac{1}{3^2}+\frac{1}{3^3}+\frac{1}{3^2}+\frac{1}{3^4}+\frac{1}{3^5}=\frac{67}{243}<1$, т.е. неравенство выполнено, но кодирование не является взаимно однозначным.

В то же время при выполнении неравенства можно построить схему кодирования, обладающую свойством взаимной однозначности, что утверждается в следующей теореме.

Теорема 20. Если числа l_i удовлетворяют неравенству Макмиллана, то существует алфавитное кодирование, Σ

$a_1 - B_1$,

...

$a_n - B_n$,

обладающее свойством префикса, такое, что $l_i = l(B_i)$.

Например, для q**=2,** l_1**=2,** l_2**=3,** l_3**=2,** l_4**=4**, l_5**=4**,

$\frac{1}{2^2}+\frac{1}{2^3}+\frac{1}{2^2}+\frac{1}{2^4}+\frac{1}{2^4}=\frac{12}{16}=\frac{3}{4}<1$ – неравенство Макмиллана выполняется. Следовательно, можно построить кодирование, обладающее свойством взаимной однозначности, обладающее свойством префикса Σ_6:

1. a_1**- 00,**
2. a_2 **– 010,**
3. a_3**- 10,**
4. a_4 **– 1110,**
5. a_5**- 1101**.

Коды с минимальной избыточностью

Пусть для входного алфавита V_a известен набор вероятностей $p_1, p_2,..., p_n$ появления символов алфавита $a_1, a_2,..., a_n$. Для заданного выходного алфавита $\{b_1, b_2,...,b_q\}$, $q \geq 2$ можно построить ряд схем алфавитного кодирования Σ

$a_1 - B_1$,

...

$a_n - B_n$,

обладающих свойством взаимной однозначности (например, префиксных). Например, можно взять слова одинаковой длины $l=]\log_q n[$. Здесь $]a[$ означает округление дробного числа до целого в большую сторону.

Определим среднюю длину элементарного кода $l_{ср} = \sum_{i=1}^{n} p_i l_i$,

где $l=l(B_i)$. Очевидно, что $l_{ср}$ показывает, во сколько раз увеличивается средняя длина слова при кодировании со схемой Σ. Определим среднюю длину элементарного кода для схем кодирования Σ_7 и Σ_8 при заданных вероятностях появления букв исходного алфавита:

Σ_7		Σ_8		p_i
$a_1 -$	00	$a_1 -$	1	$p(a_1)=0{,}4$
$a_2 -$	01	$a_2 -$	01	$p(a_2)=0{,}25$
$a_3 -$	10	$a_3 -$	000	$p(a_3)=0{,}20$
$a_4 -$	11	$a_4 -$	001	$p(a_4)=0{,}15$

Для схемы Σ_7 средняя длина слова – 2, для схемы Σ_8 - 1.95.

Для данного источника сообщений можно ввести величину l^*, где $l^* = \inf_{\Sigma} l_{ср}^{\Sigma}$.

Очевидно, что $1 \le l^* \le]\log_q n[$. Следовательно, значение l^* достигается на некоторой схеме Σ, и может быть определена как $\min\{l^{\Sigma}_{ср}\}$.

Коды, определяемые схемой Σ с $l_{ср}= l^*$ называются кодами с минимальной избыточностью, или кодами Хафмена. В силу теоремы 20 существует алфавитное кодирование с минимальной избыточностью со свойством префикса. Соотношение длин элементарных кодов и соответствующих вероятностей букв исходного алфавита даётся в следующей лемме.

Лемма. Для кода с минимальной избыточностью из условия $p_i<p_j$ следует, что $l_j \le l_i$.

На основании этой леммы строятся алгоритмы построения схем с минимальной избыточностью. Наиболее простым является алгоритм для двоичного кодирования, который рассматривается далее.

Приводимый алгоритм позволяет понять и принцип построения кодов с минимальной избыточностью для большего размера кодового алфавита.

Алгоритм построения схемы с минимальной избыточностью для двоичного кодирования

Все вероятности упорядочиваются в порядке невозрастания. Затем сливаются две минимальные вероятности. Получившиеся вероятности опять упорядочиваются в порядке невозрастания. Процедура повторяется до получения двух величин. В результате будет построено дерево кодирования, которое и разворачивается в код с минимальной избыточностью.

Например, пусть заданы вероятности букв исходного алфавита: $p(a_1)$=0,3; $p(a_2)$=0,2; $p(a_3)$=0,15; $p(a_4)$=0,1; $p(a_5)$=0,1; $p(a_6)$=0,06; $p(a_7)$=0,05; $p(a_8)$=0,04.

В таблице приводится пример построения такого кода:

0,3	0,3	0,3	0,3	0,3	***0,4***	***0,6***	***0***	***1***	00	00	00	00	00
0,2	0,2	0,2	0,2	***0,3***	0,3	0,4	1	00	***01***	10	10	10	10
0,15	0,15	0,15	***0,2***	0,2	0,3			01	10	***11***	010	010	010
0,1	0,1	***0,15***	0,15	0,2					11	010	***011***	110	110
0,1	0,1	0,1	0,15							011	110	111	111
0,06	***0,09***	0,1									111	***0110***	0111
0,05	0,06											0111	01100
0,04													01101

Левая часть таблицы соответствует слиянию вероятностей до получения двух вероятностей. Правая часть таблицы соответствует построению префиксного кода с минимальной избыточностью. Курсивом выделены вероятности, полученные в результате слияния, и коды, для которых производится расщепление вершин.

Кодовое дерево для построенной схемы представлено на рис.56. У концевых вершин дерева указаны вероятности соответствующих этим кодам букв (а не сами буквы, как обычно указывается в кодовых деревьях, поскольку в данном случае важны именно вероятности).

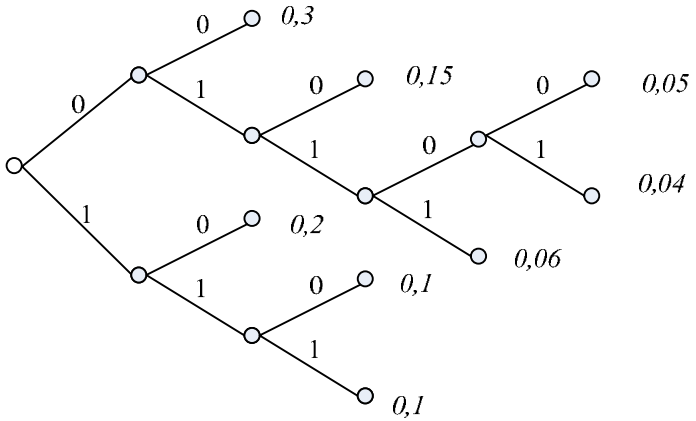


Рис. 56. Кодовое дерево для кода Хафмена

Самокорректирующиеся коды

Рассматриваются коды, позволяющие исправлять ошибки, которые могут появиться при появлении помех в канале связи. Пусть кодовый алфавит состоит из двух букв $V_b=\{0, 1\}$. $A=\{A_1, A_2, \dots A_n\}$- множество различных слов фиксированной длины m ($n \le 2^m$), $A_i \in V_b^+$.

Рассматривается единичный m-мерный куб как метрическое пространство. Расстояние $\rho(b', b'')$ между двумя векторами $b'=(b'_1 b'_2 \dots b'_m)$ и $b''=(b''_1 b''_2 \dots b''_m)$ определяется следующим образом : $\rho(b', b'') = \sum_{i=1}^{m} \left| b'_i - b''_i \right|$, т.е. число координат, в которых различаются вектора.

Если минимальное расстояние между различными векторами равно единице, то некоторые коды могут в результате появления ошибки превратиться в другие правильные коды, и единичная ошибка замечена не будет. Если же минимальное расстояние между векторами равно 2, то единичная ошибка будет замечена, поскольку полученный код не будет соответствовать ни одному из

допустимых, но исправить её не удастся, так как неизвестно, какой именно из ближайших кодов был передан. Поэтому для того, чтобы корректировать одну ошибку в коде, минимальное расстояние между кодами должно быть равно 3. Для коррекции p ошибок минимальное расстояние между двумя кодами должно быть $\geq 2p+1$.

Коды Хемминга разработаны для обнаружения и исправления ошибок. Структура кода такова: все разряды кода, номера которых являются степенями двойки, являются контрольными, остальные разряды – смысловыми. Сначала заполняются смысловые разряды кода, затем строятся контрольные разряды. Контрольные разряды строятся следующим образом:

$b_1=b_3 \oplus b_5 \oplus b_7 \oplus \ldots \oplus b_{2n+1} \oplus \ldots$, т.е. суммируются все нечётные разряды, кроме первого,

$b_2=b_3 \oplus b_6 \oplus b_7 \oplus \ldots \oplus b_k \oplus \ldots$, где k – числа, имеющие единицу во втором разряде двоичного кода (кроме самой двойки),

$b_4=b_5 \oplus b_6 \oplus b_7 \oplus \ldots \oplus b_k \oplus \ldots$, где k – числа, имеющие единицу в третьем разряде двоичного кода(кроме самой 4), ...

$b_n=\ldots \oplus b_k \oplus \ldots$ где $n=2^j$, k – числа, имеющие единицу в j-м разряде двоичного кода (кроме самого n) .

Код Хемминга для $m=4$ обеспечивает расстояние между элементарными кодами, равное 3 и исправление единичной ошибки. Рассмотрим построение контрольных разрядов для $m=4$:

$b_1=b_3 \oplus b_5 \oplus b_7$,

$b_2=b_3 \oplus b_6 \oplus b_7$,

$b_4=b_5 \oplus b_6 \oplus b_7$.

Для кода Хемминга при одной ошибке адрес ошибочного бита определяется следующим образом:

$s_1=b_1 \oplus b_3 \oplus b_5 \oplus b_7 \oplus \ldots \oplus b_{2n+1} \oplus \ldots$

$s_2=b_2 \oplus b_3 \oplus b_6 \oplus b_7 \oplus \ldots \oplus b_k \oplus \ldots$, где k – числа, имеющие единицу во втором разряде двоичного кода,

$s_4=b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus \ldots \oplus b_k \oplus \ldots$, где k – числа, имеющие единицу в третьем разряде двоичного кода,...

$s_n=\ldots \oplus b_k \oplus \ldots$ где $n=2^j$, k – числа, имеющие единицу в j-м разряде двоичного кода.

$S'=s_n...s_4s_2s_1$ – число, которое определяет адрес ошибочного бита.

Если $s_1=0$, то S' не принадлежит первой последовательности, если $s_1=1$, то S' принадлежит первой последовательности. Аналогично можно рассмотреть все остальные разряды.

Если $S'=0$, то ошибки нет, иначе бит с адресом S' заменяется на своё дополнение.

Пример.

Пусть исходный код: **0110**. Символы кода помещаются последовательно в 3-й, 5-й, 6-й и 7-й разряды:

. . 0 .110

Строим код Хемминга:

$b_1=b_3 \oplus b_5 \oplus b_7=0 \oplus 1 \oplus 0=1,$

$b_2=b_3 \oplus b_6 \oplus b_7=0 \oplus 1 \oplus 0=1,$

$b_4=b_5 \oplus b_6 \oplus b_7=1 \oplus 1 \oplus 0=0$.

Таким образом, получен код Хемминга **1100110**.

Рассмотрим случай внесения единичной ошибки. Изменим четвёртый разряд, получим код **1101110**. Считаем S':

$s_1=b_1 \oplus b_3 \oplus b_5 \oplus b_7=1+0+1+0=0,$

$s_2=b_2 \oplus b_3 \oplus b_6 \oplus b_7=1+0+1+0=0$

$s_4=b_4 \oplus b_5 \oplus b_6 \oplus b_7=1+1+1+0=1$, таким образом, $S'=100$. Значит, ошибка появилась в четвёртом разряде. Заменяем значение этого разряда, получаем: **1100110** – правильный код.

Построение автоматов, распознающих префиксные коды

Если дан префиксный код, то всегда можно построить автомат, распознающий этот префиксный код. При этом на входе будет цепочка символов кодирующего алфавита, на выходе – декодированное сообщение.

Пусть задана префиксная схема кодирования Σ:

$a_1 - B_1,$

…

$a_к - B_к$

Поскольку схема кодирования является префиксной, то для данной схемы существует кодовое дерево. Кодовое дерево префикс-

ной схемы кодирования можно преобразовать в граф переходов автомата Мили.

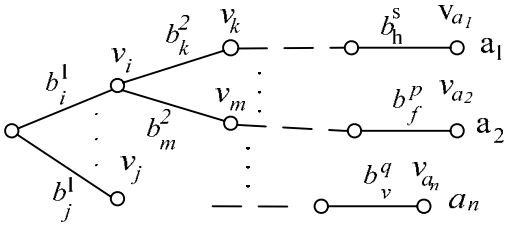


Рис. 57. Кодовое дерево для префиксной схемы кодирования

Алгоритм построения графа переходов декодирующего автомата

Пусть дано кодовое дерево для префиксной схемы кодирования Σ.

1. Начальной вершине дерева сопоставляем вершину q_0.

2. Всем внутренним вершинам дерева сопоставляем вершины графа переходов (внутренней вершине v_i дерева сопоставляем вершину q_i графа).

3. Каждому ребру, ведущему из вершины v_i к внутренней вершине дерева v_j, помеченную буквой b_k, сопоставляем дугу из вершины q_i в вершину q_j, и помечаем её (b_k, –).

4. Ребру из внутренней вершины дерева v_i в висячую вершину, сопоставленную букве a_s исходного алфавита, сопоставим дугу из вершины q_i в начальную вершину графа переходов q_0 и помечаем её (b_k, a_s).

Полученный граф переходов будет соответствовать декодирующему автомату для префиксного кода, заданного схемой Σ. По графу переходов автомата, очевидно, можно полностью описать автомат.

Рассмотрим пример построения графа переходов декодирующего автомата. Пусть задана префиксная схема кодирования Σ_9**:**

$a_1 - cc$**,**

$a_2 - cbc$**,**

$a_3 - cbb$**,**

$a_4 - b$**.**

Соответствующее кодовое дерево с помеченными внутренними вершинами представлено на рис. 58.

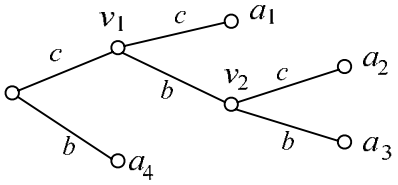


Рис. 58. Кодовое дерево для схемы кодирования Σ_9

Построим граф переходов распознающего автомата для префиксной схемы кодирования Σ_9.

1. Множество вершин графа переходов состоит из 3-х вершин q_0 (соответствует начальной вершине), q_1 (соответствует вершине v_1), q_2 (соответствует вершине v_2).

2. Множество дуг состоит из шести дуг, каждая из которых соответствует ветви дерева. Поскольку существует элементарный код длиной 1, в графе переходов присутствует петля.

Построенный граф переходов автомата представлен на рис.59.

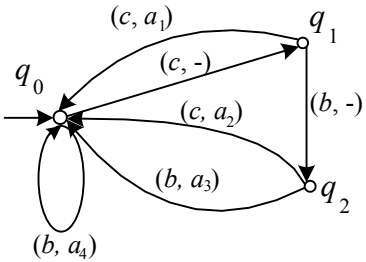


Рис.59. Граф переходов декодирующего автомата для схемы Σ_9

Вопросы и упражнения

1. Как проверить, обладает ли код свойством префикса?
2. Всегда ли префиксный код обеспечивает однозначность декодирования?
3. Выполнение какого условия обеспечивает возможность построить однозначный алфавитный код с заданными длинами кодов?
4. Можно ли построить однозначный код в алфавите из трёх букв с длинами элементарных кодов 1, 1, 2, 2,2?
5. Что такое коды с минимальной избыточностью?
6. Может ли в коде с минимальной избыточностью длина элементарного кода для буквы с вероятностью 0,3 быть равна длине элементарного кода с вероятностью 0,05?
7. Построить код с минимальной избыточностью для алфавита со следующими частотами символов: 0,3; 0,25; 0,1; 0,1; 0,07; 0,06; 0,05; 0,04; 0,03.
8. Что такое самокорректирующиеся коды?
9. Как определяется расстояние между кодами в равномерном кодировании?
10. Построить код Хемминга, исправляющий единичную ошибку, для сообщения 0101.

11. Проверить наличие ошибки в коде Хемминга, исправляющем единичную ошибку **1110111**. Если есть ошибка, исправить её. Восстановить смысловую часть сообщения.

12. Может ли быть построен самокорректирующийся код с минимальным расстоянием в 1 между кодами?

13. Каково должно быть минимальное расстояние между кодами в равномерном кодировании для исправления двух ошибок? Трёх?

14. Построить автомат для распознавания кода Хафмена, кодовое дерево для которого представлено на рис. 56.

Список литературы

1. Яблонский С.В. Введение в дискретную математику. М.: Высшая школа, 2001.
2. Кузнецов О.П., Адельсон- Вельский Г.М. Дискретная математика для инженера М.: Энергоатомиздат, 1988
3. Гаврилов Г.П., Сапоженко А.А.Задачи и упражнения по курсу дискретной математики. М.: Наука, 1992.
4. Ахо А., Ульман Дж. Теория синтаксического анализа перевода и компиляции. Т. 1,2 . Пер. с англ. М.: Мир 1978.
5. Сергиевский Г.М., Короткова М.А. Введение в математическую лингвистику и теорию автоматов. Конспект лекций М.: МИФИ, 2004.