

Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики



**В.Н. Пильщиков, В.Г. Абрамов,
А.А. Вылиток, И.В. Горячая**

**Машина Тьюринга и алгоритмы Маркова.
Решение задач**

Учебно-методическое пособие

Москва, 2016

УДК 681.325.5
ББК 22.18
ПЗ2

Пильщиков В.Н., Абрамов В.Г., Вылиток А.А., Горячая И.В. Машина Тьюринга и алгоритмы Маркова. Решение задач. (Учебно-методическое пособие) 2-е исправленное и дополненное издание - М.: МГУ, 2016. – 72 с.

Издательский отдел факультета ВМК МГУ (лицензия ИД №05899 от 24.09.2001 г.)

Пособие посвящено решению задач по теме «Введение в теорию алгоритмов», изучаемой на первом курсе факультета ВМК МГУ в рамках дисциплины «Алгоритмы и алгоритмические языки». Это задачи на составление алгоритмов в виде машины Тьюринга и нормальных алгоритмов Маркова, а также задачи теоретического характера.

В пособии приводятся необходимые сведения по теории алгоритмов, подробно объясняются типичные приёмы решения задач и предлагается большой набор задач для самостоятельного решения.

Пособие рассчитано на студентов первого курса факультета ВМК МГУ и преподавателей, ведущих семинарские занятия по программированию.

Ключевые слова: машина Тьюринга, нормальные алгоритмы Маркова, эквивалентность алгоритмов, композиция алгоритмов, алгоритмическое решение задач

V.N. Pilschikov, V.G. Abramov, A.A. Vylitok, I.V. Goryachaya Turing machine and Markov algorithms. Problem solving and brief theory. (Study guide)

The guide deals with problem solving on: “Introduction to Algorithm theory” studied during the first year in Moscow State University on Computational Mathematics and Cybernetics faculty as a part of “Algorithms and Algorithmic Languages” course. These are the problems to compose algorithms in the way of Turing machine and normal Markov algorithms as well as theoretical-like tasks.

The guide provides necessary information on algorithms theory, detailed description of typical solutions and a large set of problems for self-study.

The present edition is based on the edition of 2006 with correction of the found mistakes, expanded set of problems and theoretical part.

The guide is designed for the first year students of Computational Mathematics and Cybernetics faculty of the Moscow State University and teaching staff conducting seminars on programming subject.

Key words: Turing machine, Markov's Normal Algorithms, equivalence of algorithms, composition of algorithms, algorithmic problem solving

Рецензенты:

доцент Баула В.Г.

доцент Корухова Л.С.

Печатается по решению Редакционно-издательского совета факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова.

ISBN 978-5-89407-563-1

© Издательский отдел факультета
вычислительной математики и кибернетики
МГУ им. М.В. Ломоносова, 2016

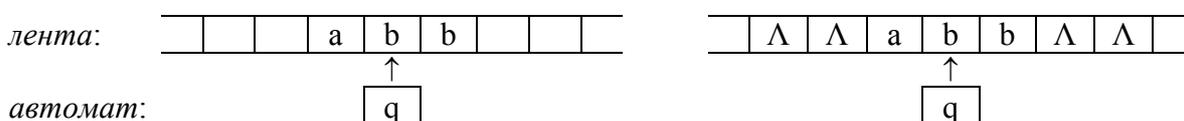
1. Машина Тьюринга

В разделе рассматриваются задачи на составление алгоритмов для машины Тьюринга. Приводится краткое описание этой машины, на примерах объясняются основные приёмы составления таких алгоритмов и предлагаются задачи для самостоятельного решения.

1.1 Краткое описание машины Тьюринга

Структура машины Тьюринга

Машина Тьюринга (МТ) состоит из двух частей – *ленты* и *автомата* (см. слева):



Лента используется для хранения информации. Она бесконечна в обе стороны и разбита на клетки, которые никак не нумеруются и не именованы. В каждой клетке может быть записан один символ или ничего не записано. Содержимое клетки может меняться – в неё можно записать другой символ или стереть находящийся там символ.

Договоримся пустое содержимое клетки называть *символом «пусто»* и обозначать знаком Λ («лямбда»). В связи с этим изображение ленты, показанное на рисунке справа, такое же, как и на рисунке слева. Данное соглашение удобно тем, что операцию стирания символа в некоторой клетке можно рассматривать как запись в эту клетку символа Λ , поэтому вместо длинной фразы «записать символ в клетку или стереть находящийся там символ» можно говорить просто «записать символ в клетку».

Автомат – это активная часть МТ. В каждый момент он размещается под одной из клеток ленты и видит её содержимое; это *видимая клетка*, а находящийся в ней символ – *видимый символ*; содержимое же соседних и других клеток автомат не видит. Кроме того, в каждый момент автомат находится в одном из *состояний*, которые будем обозначать буквой q с номерами: q_1 , q_2 и т.п. Находясь в некотором состоянии, автомат выполняет какую-то определённую операцию (например, перемещается направо по ленте, заменяя все символы b на a), находясь в другом состоянии – другую операцию.

Пару из видимого символа (S) и текущего состояния автомата (q) будем называть **конфигурацией** и обозначать $\langle S, q \rangle$.

Автомат может выполнять три элементарных действия: 1) записывать в видимую клетку новый символ (менять содержимое других клеток автомат не может); 2) сдвигаться на одну клетку влево или вправо («перепрыгивать» сразу через несколько клеток автомат не может); 3) переходить в новое состояние. Ничего другого делать автомат не умеет, поэтому все более сложные операции так или иначе должны быть сведены к этим трём элементарным действиям.

Такт работы машины Тьюринга

МТ работает **тактами** (*по шагам*), которые выполняются один за другим. На каждом такте автомат МТ выполняет три следующих действия, причем обязательно в указанном порядке:

1) записывает некоторый символ S' в видимую клетку (в частности, может быть записан тот же символ, что и был в ней, тогда содержимое этой клетки не меняется);

2) сдвигается на одну клетку влево (обозначение – L , от *left*), либо на одну клетку вправо (обозначение – R , от *right*), либо остается неподвижным (обозначение – N).

3) переходит в некоторое состояние q' (в частности, может остаться в прежнем состоянии).

Формально действия одного такта будем записывать в виде тройки:

$$S', [L,R,N], q'$$

где конструкция с квадратными скобками означает возможность записи в этом месте любой из букв L , R или N . Например, такт $*L, q\delta$ означает запись символа $*$ в видимую клетку, сдвиг на одну клетку влево и переход в состояние $q\delta$. Запись такта для конфигурации называют **командой** (машины Тьюринга).

Программа для машины Тьюринга

Сама по себе МТ ничего не делает. Для того чтобы заставить её работать, надо написать для неё **программу**. Эта программа записывается в виде следующей таблицы:

	S_1	S_2	...	S_i	...	S_n	Λ
q_1							
...							
q_j				$S', [L, R, N], q'$			
...							
q_m							

Слева перечисляются все состояния, в которых может находиться автомат, сверху – все символы (в том числе и Λ), которые автомат может видеть на ленте. (Какие именно символы и состояния указывать в таблице – определяет автор программы.) На пересечениях же (в ячейках таблицы) указываются те такты, которые должен выполнить автомат, когда он находится в соответствующем состоянии и видит на ленте соответствующий символ.

В целом таблица определяет действия МТ при всех возможных конфигурациях и тем самым полностью задаёт поведение МТ. Описать алгоритм в виде МТ – значит предъявить такую таблицу.

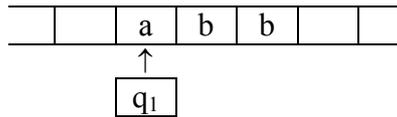
(Замечание. Часто МТ определяют как состоящую из ленты, автомата и программы, поэтому при разных программах получаются разные МТ. Мы же будем считать, в духе современных компьютеров, что МТ одна, но она может выполнять разные программы.)

Правила выполнения программы

К началу выполнения программы машина Тьюринга находится в **начальной** конфигурации. Эта начальная конфигурация определена следующим образом.

Во-первых, на ленте записано **входное слово**, к которому будет применена программа. Входное слово – это конечная последовательность символов, записанных в соседних клетках ленты; внутри входного слова пустых клеток быть не должно, а слева и справа от него должны быть только пустые клетки. Пустое входное слово означает, что все клетки ленты пусты.

Во-вторых, автомат установлен в состояние q_1 (указанное в таблице первым) и размещен под его первым (самым левым) символом входного слова:



Если входное слово пустое, то автомат может смотреть в любую клетку, т.к. все они пусты.

После этих предварительных действий начинается выполнение программы. В таблице отыскивается ячейка на пересечении первой строки (т.к. автомат находится в состоянии q_1) и того столбца, который соответствует первому символу входного слова (это необязательно левый столбец таблицы), и выполняется такт, указанный в этой ячейке. В результате автомат окажется в новой конфигурации. Теперь такие же действия повторяются, но уже для новой конфигурации: в таблице отыскивается ячейка, соответствующая состоянию и символу этой конфигурации, и выполняется такт из этой ячейки. И так далее.

Когда завершается выполнение программы? Введём понятие **такта останова**. Это такт, который ничего не меняет: автомат записывает в видимую клетку тот же символ, что и был в ней раньше, не сдвигается и остается в прежнем состоянии, т.е. это такт S, N, q для конфигурации $\langle S, q \rangle$. Попад на такт останова, МТ, по определению, останавливается, завершая свою работу.

В целом возможны два исхода работы МТ над входным словом:

1) Первый исход – «хороший»: это когда в какой-то момент МТ останавливается (попадает на такт останова). В таком случае говорят, что МТ **применима** к заданному входному слову. А то слово, которое к этому моменту получено на ленте, считается **выходным словом**, т.е. результатом работы МТ, ответом.

Будем считать, что в момент останова должны выполняться следующие условия:

– внутри выходного слова не должно быть пустых клеток (отметим, что во время выполнения программы внутри обрабатываемого слова пустые клетки могут быть, но в конце их уже не должно остаться);

– автомат обязан остановиться под одним из символов выходного слова (под каким именно – не играет роли), а если слово пустое – под любой клеткой ленты.

2) Второй исход – «плохой»: это когда МТ заикливается, никогда не попадая на такт останова (например, автомат на каждом шаге сдвигается вправо и потому не может остановиться, т.к. лента

бесконечна). В этом случае говорят, что МТ *неприменима* к заданному входному слову. Ни о каком результате при таком исходе не может идти и речи.

Отметим, что один и тот же алгоритм (программа МТ) может быть применимым к одним входным словам (т.е. останавливаться) и неприменимым к другим (т.е. заикливаться). Таким образом, применимость/неприменимость зависит не только от самого алгоритма, но и от входного слова.

На каких входных словах алгоритм должен останавливаться? На, так сказать, хороших словах, т.е. на тех, которые относятся к допустимым исходным данным решаемой задачи, для которых задача осмысленна. Но на ленте могут быть записаны любые входные слова, в том числе и те, для которых задача не имеет смысла; на таких словах поведение алгоритма не фиксируется, он может остановиться (при любом результате), а может и заиклиться.

Соглашения для сокращения записи

Договоримся о некоторых соглашениях, сокращающих запись программы для МТ.

1) Если в такте не меняется видимый символ, или автомат не сдвигается, или не меняется состояние автомата, то в соответствующей позиции такта мы не будем ничего писать.

Например, при конфигурации $\langle a, q1 \rangle$ следующие записи тактов эквивалентны:

$$\begin{aligned} a, R, q3 &\equiv \quad , R, q3 && \text{(но не } \Lambda, R, q3 \text{ !!)} \\ b, N, q2 &\equiv b, \quad , q2 \\ a, L, q1 &\equiv \quad , L, \\ a, N, q1 &\equiv \quad , \quad , && \text{(это такт останова)} \end{aligned}$$

Замечание. Запяты в тактах желательно не опускать, т.к. иначе возможна путаница, если среди символов на ленте могут встретиться буквы L и R .

2) Если надо указать, что после выполнения некоторого такта МТ должна остановиться, то в третьей позиции этого такта будем писать знак «!». Например, такт $b, L, !$ означает следующие действия: запись символа b в видимую клетку ленты, сдвиг влево и останов.

Формально можно считать, что в программе МТ имеется состояние с названием $!$, во всех ячейках которого записаны такты останова. При

этом, однако, такую строку явно не выписывают, а лишь подразумевают.

3) Если заранее известно, что в процессе выполнения программы не может появиться некоторая конфигурация, тогда, чтобы подчеркнуть это явно, будем в соответствующей ячейке таблицы рисовать крестик.

Эти соглашения необязательны, но они сокращают запись программы и упрощают её восприятие.

1.2 Примеры на составление программ для МТ

Рассмотрим примеры на составление программ для МТ, чтобы продемонстрировать некоторые типичные приёмы программирования на МТ.

Для сокращения формулировки задач введём следующие два соглашения:

- буквой P будем обозначать входное слово;
- буквой A будем обозначать алфавит входного слова, т.е. набор тех символов, из которых и только которых может состоять P (отметим, однако, что в промежуточных и выходном словах могут появляться и другие символы).

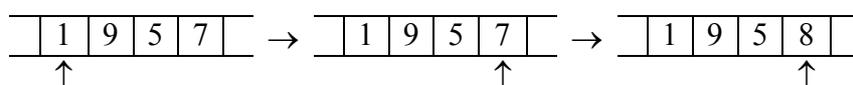
Пример 1 (перемещение автомата, замена символов)

$A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Пусть P – непустое слово; значит, P – это последовательность из десятичных цифр, т.е. запись неотрицательного целого числа в десятичной системе. Требуется получить на ленте запись числа, которое на 1 больше числа P .

Решение

Для решения этой задачи предлагается выполнить следующие действия:

1. Перегнать автомат под последнюю цифру числа.
2. Если это цифра от 0 до 8, то заменить её цифрой на 1 больше и остановиться; например:



т.к. должен выполнить то же самое действие – увеличить на 1 видимую цифру. Если же автомат сдвинулся влево, а в видимой клетке нет цифры (а есть «пусто»), то он записывает сюда 1 и останавливается.

Отметим, что для пустого входного слова наша задача не определена, поэтому на этом слове МТ может вести себя как угодно. В нашей программе, например, при пустом входном слове МТ останавливается и выдает ответ 1.

Выше мы привели запись программы в полном, несокращённом виде. Теперь же приведём запись программы в сокращённом, более наглядном виде, при этом справа дадим краткое пояснение действий, которые реализуются в соответствующих состояниях автомата:

	0	1	2	3	4	5	6
q1	,R,	,R,	,R,	,R,	,R,	,R,	
q2	1, ,!	2, ,!	3, ,!	4, ,!	5, ,!	6, ,!	

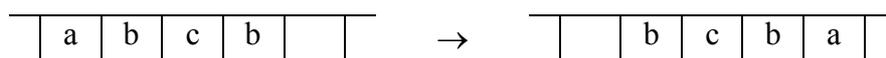
	6	7	8	9	Λ	<i>комментарий</i>
q1	,R,	,R,	,R,	,R,	,L,q2	под последнюю цифру
q2	7, ,!	8, ,!	9, ,!	0,L,	1, ,!	видимая цифра + 1

Именно так мы и будем в дальнейшем записывать программы.

Пример 2 (анализ символов)

$A = \{a, b, c\}$. Перенести первый символ непустого слова P в его конец.

Например:



Решение

Для решения этой задачи предлагается выполнить следующие действия:

1. Запомнить первый символ слова P , а затем стереть этот символ.
2. Перегнать автомат вправо под первую пустую клетку за P и записать в неё запомненный символ.

Как «бегать» вправо, мы уже знаем из предыдущего примера. А вот как запомнить первый символ? Ведь в МТ нет другого запоминающего устройства, кроме ленты, а запоминать символ в какой-то клетке на ленте бессмысленно: как только автомат сдвинется влево или вправо от этой клетки, он тут же забудет данный символ. Что делать?

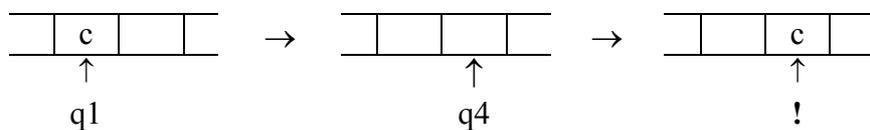
Выход здесь таков – надо использовать разные состояния автомата. Если первый символ – это a , то надо перейти в состояние $q2$, в котором автомат бежит вправо и записывает в конце a . Если же первым был символ b , тогда надо перейти в состояние $q3$, где делается всё то же самое, только в конце записывается символ b . Если же первым был символ c , тогда переходим в состояние $q4$, в котором автомат дописывает за входным словом символ c . Следовательно, то, каким был первый символ, мы фиксируем переводом автомата в разные состояния. Это типичный приём при программировании на МТ.

С учётом сказанного программа будет такой:

	a	b	c	Λ	<i>комментарий</i>
q1	$\Lambda, R, q2$	$\Lambda, R, q3$	$\Lambda, R, q4$	$, R,$	анализ 1-го символа, удаление его, разветвление
q2	$, R,$	$, R,$	$, R,$	$a, ,!$	запись a справа
q3	$, R,$	$, R,$	$, R,$	$b, ,!$	запись b справа
q4	$, R,$	$, R,$	$, R,$	$c, ,!$	запись c справа

Рассмотрим поведение этой программы на входных словах, содержащих не более одного символа. При пустом слове, которое является «плохим» для задачи, программа заикнется – автомат, находясь в состоянии $q1$ и попадая всё время на пустые клетки, будет бесконечно перемещаться вправо. (Конечно, в этом случае программу можно было бы остановить, но мы специально сделали заикливание, чтобы продемонстрировать такую возможность.)

Если же во входном слове ровно один символ, тогда автомат сотрёт этот символ, сдвинется на одну клетку вправо и запишет в неё данный символ:



Таким образом, слово из одного символа попросту сдвинется на клетку вправо. Это допустимо. Ведь клетки ленты не нумерованы, поэтому местоположение слова на ленте никак не фиксируется и перемещение слова влево или вправо заметить нельзя. В связи с этим не требуется, чтобы выходное слово обязательно находилось в том же месте, где было входное слово, результат может оказаться и левее, и правее этого места.

Пример 3 (сравнение символов, стирание слова)

$A=\{a,b,c\}$. Если первый и последний символы (непустого) слова P одинаковы, тогда это слово не менять, а иначе заменить его на пустое слово.

Решение

Для решения этой задачи предлагается выполнить следующие действия:

1. Запомнить первый символ входного слова, не стирая его.
2. Переместить автомат под последний символ и сравнить его с запомненным. Если они равны, то больше ничего не делать.
3. В противном случае уничтожить всё входное слово.

Как запоминать символ и как перегонять автомат под последний символ слова, мы уже знаем из предыдущих примеров. Стирание же входного слова реализуется заменой всех его символов на символ Λ . При этом, раз уж автомат оказался в конце слова, будем перемещать автомат справа налево до первой пустой клетки.

Эти действия описываются следующей программой для МТ (напомним, что крестик в ячейке таблицы означает невозможность появления соответствующей конфигурации при выполнении программы):

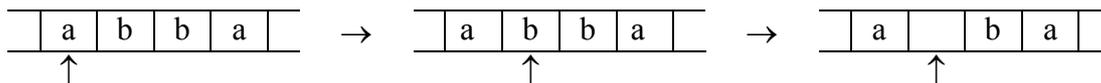
	a	b	c	Λ	<i>комментарий</i>
q1	,,q2	,,q4	,,q6	,,!	анализ 1-го символа, разветвление
q2	,R,	,R,	,R,	, L,q3	идти к последнему символу при 1-м символе a
q3	,,!	,, q8	,, q8	×	сравнить последний символ с a , не равны – на q8 (стереть P)
q4	,R,	,R,	,R,	, L,q5	аналогично при 1-м символе b
q5	,, q8	,,!	,, q8	×	
q6	,R,	,R,	,R,	, L,q7	аналогично при 1-м символе c
q7	,, q8	,, q8	,,!	×	
q8	$\Lambda,L,$	$\Lambda,L,$	$\Lambda,L,$,,!	стереть всё слово, двигаясь справа налево

Пример 4 (удаление символа из слова)

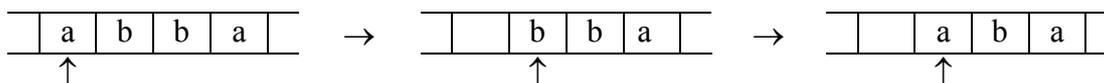
$A=\{a,b\}$. Удалить из слова P его второй символ, если такой есть.

Решение

Казалось бы, эту задачу решить просто: надо сдвинуть автомат под клетку со вторым символом и затем очистить эту клетку:



Однако напомним, что внутри выходного слова не должно быть пустых клеток. Поэтому после удаления второго символа надо «сжать» слово, перенеся первый символ на одну клетку вправо. Для этого автомат должен вернуться к первому символу, запомнить его и стереть, а затем, снова сдвинувшись вправо, записать его в клетку, где был второй символ. Однако начальный «поход» вправо ко второму символу, чтобы его стереть, и последующий возврат к первому символу являются лишними действиями: какая разница – переносить первый символ в пустую клетку или в клетку с каким-то символом? Поэтому сразу запоминаем первый символ, стираем его и записываем вместо второго символа:



В виде программы для МТ всё это записывается так:

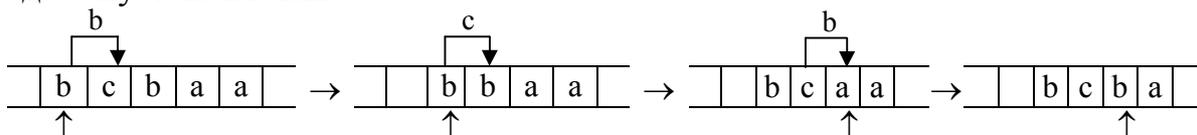
	a	b	Λ	комментарий
q1	Λ,R,q2	Λ,R,q3	,,!	анализ и удаление 1-го символа, разветвление
q2	,,!	a,,!	a,,!	замена 2-го символа на a
q3	b,,!	,,!	b,,!	замена 2-го символа на b

Пример 5 (сжатие слова)

$A = \{a, b, c\}$. Удалить из слова P первое вхождение символа a , если такое есть.

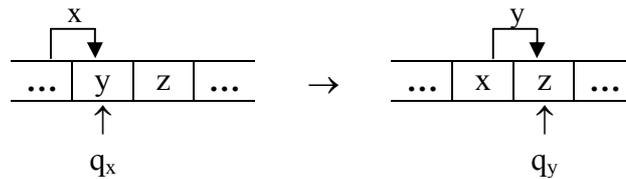
Решение

В предыдущем примере мы переносили на позицию вправо только один символ. В данном же примере мы будем в цикле переносить вправо все начальные символы b и c входного слова – до первого символа a или до пустой клетки:



Центральный момент здесь – как перенести символ x из левой клетки в очередную клетку, где находится некоторый символ y , чтобы затем этот символ y можно было перенести в правую клетку? Если через

q_x обозначить состояние, в котором в видимую клетку надо записать символ x , находившийся ранее в клетке слева, тогда это действие можно изобразить так:



Для этого предлагается выполнить такт x,R,q_y , в котором объединены следующие три действия: во-первых, в видимую клетку записывается символ x , взятый из клетки слева; во-вторых, автомат сдвигается вправо – под клетку, в которую затем надо будет записать только что заменённый символ y ; в-третьих, автомат переходит в состояние q_y , в котором он и будет выполнять эту запись.

Повторение таких тактов в цикле и приведёт к сдвигу вправо на одну позицию начальных символов входного слова. Этот цикл должен закончиться, когда в очередной клетке окажется символ a или Λ ($y=a$ или $y=\Lambda$), а в начале цикла можно считать, что на место первого символа слева переносится символ «пусто» ($x=\Lambda$). В итоге получается следующая программа для МТ:

	a	b	c	Λ	комментарий
q1	$\Lambda,R,! $	$\Lambda,R,q2$	$\Lambda,R,q3$	$,,! $	q_Λ : стереть 1-й символ и перенести его вправо
q2	$b,;! $	$,R,$	$b,R,q3$	$b,;! $	q_b : запись b и запоминание стертого символа
q3	$c,;! $	$c,R,q2$	$,R,$	$c,;! $	q_c : запись c и запоминание стертого символа

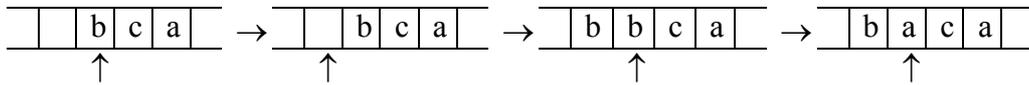
В этой программе следует обратить внимание на такт $\Lambda,R,!$, который выполняется в конфигурации $\langle a, q1 \rangle$, т.е. когда первым символом входного слова является a . Ясно, что надо просто стереть этот символ и остановиться. Однако в этом такте указан ещё и сдвиг вправо. Зачем? Напомним, что в момент останова автомат должен находиться под выходным словом (под любым его символом), поэтому мы и сдвигаем автомат с пустой клетки на клетку с первым символом выходного слова, который во входном слове был вторым.

Пример 6 (вставка символа в слово)

$A=\{a,b,c\}$. Если P – непустое слово, то за его первым символом вставить символ a .

Решение

Ясно, что между первым и вторым символами слова P надо освободить клетку для нового символа a . Для этого надо перенести на одну позицию влево первый символ (на старом месте его можно пока не удалять), а затем, вернувшись на старое место, записать символ a :



Перенос символа на одну позицию влево аналогичен переносу символа вправо, о чём говорилось в двух предыдущих примерах, поэтому приведем программу для МТ без дополнительных комментариев. Отметим лишь, что в состояниях q_2 , q_3 и q_4 автомат может видеть только пустую клетку, а в состоянии q_5 он обязательно видит первый символ входного слова, но не пустую клетку.

	a	b	c	Λ	комментарий
q_1	,L, q_2	,L, q_3	,L, q_4	,,!	анализ 1-го символа для переноса его влево
q_2	×	×	×	a,R, q_5	приписать a слева
q_3	×	×	×	b,R, q_5	приписать b слева
q_4	×	×	×	c,R, q_5	приписать c слева
q_5	,,!	a,!	a,!	×	заменить бывший 1-й символ на a

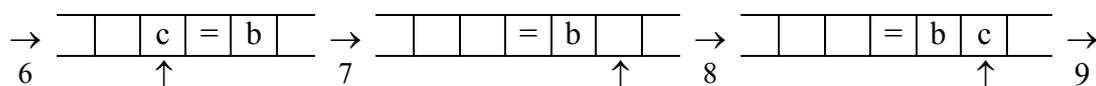
Пример 7 (раздвижка слова)

$A = \{a, b, c\}$. Вставить в слово P символ a за первым вхождением символа c , если такое есть.

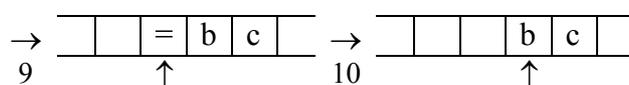
Решение

Просматриваем входное слово слева направо до пустой клетки или до первого символа c . В первом случае c не входит в P , поэтому ничего не делаем. Во втором случае надо освободить место для вставляемого символа a , для чего сдвигаем начало слова P (от первого символа до найденного символа c) на одну позицию влево. При этом осуществляем такой сдвиг справа налево – от символа c к началу слова, раз уж автомат находится под этим символом. Кроме того, чтобы затем не возвращаться к освободившейся позиции, начинаем этот сдвиг с записи a вместо найденного символа c . Поскольку этот циклический сдвиг влево реализуется аналогично сдвигу в цикле вправо из примера 5, то не будем пояснять его, а сразу приведём программу для МТ:

Снова возвращаемся налево к тому символу, который стал первым во входном слове, и повторяем те же самые действия, но уже по отношению к этому символу (см. шаги 6-9).



4. Этот цикл завершается, когда при возврате налево мы увидим в качестве первого символа знак =. Это признак того, что мы полностью просмотрели входное слово и перенесли все его символы, отличные от a , в формируемое справа выходное слово. Надо этот знак стереть, сдвинуться вправо под выходное слово и остановиться (см. шаг 10).



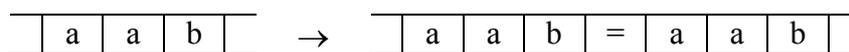
С учётом всего сказанного и строим программу для МТ. При этом отметим, что помимо символов a , b и c в процессе решения задачи на ленте появляется знак =, поэтому в таблице должен быть предусмотрен столбец и для этого знака.

	a	b	c	=	Λ	комментарий
q1	,R,	,R,	,R,	×	=, q2	записать справа знак =
q2	,L,	,L,	,L,	,L,	,R, q3	влево к 1-му символу слова
q3	Λ, R,	Λ, R, q4	Λ, R, q5	Λ, R, !	×	анализ и удаление его, разветвление
q4	,R,	,R,	,R,	,R,	b, q2	запись b справа, возврат налево (в цикл)
q5	,R,	,R,	,R,	,R,	c, q2	запись c справа, возврат налево (в цикл)

Пример 9 (фиксирование места на ленте)

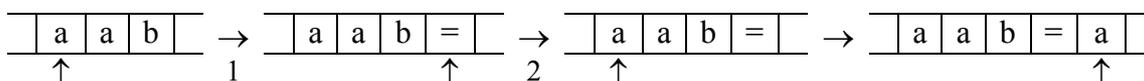
$A = \{a, b\}$. Удвоить слово P , поставив между ним и его копией знак =.

Например:



Решение

Эта задача решается аналогично предыдущей: в конец входного слова записываем знак =, затем возвращаемся к началу слова и в цикле все его символы (в том числе и a) копируем в пустые клетки справа:



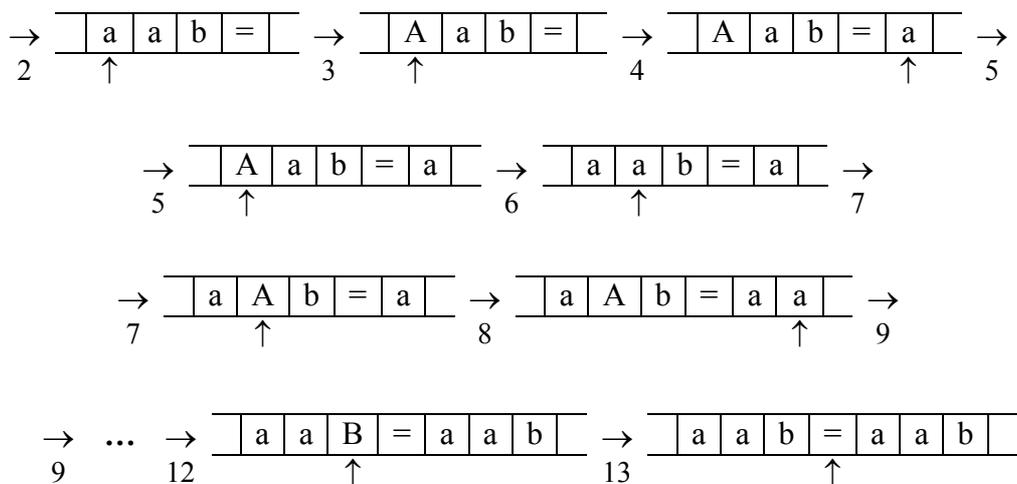
Однако есть и отличие: копируемые символы входного слова не удаляются, и это приводит к следующей проблеме. Записав справа копию очередного символа, мы затем должны вернуться к входному слову в позицию этого символа и потом сдвинуться вправо к следующему символу, чтобы скопировать уже его. Но как узнать, в какую позицию входного слова надо вернуться? Например, откуда мы знаем в нашем примере, что после копирования первого символа a мы должны вернуться именно к первому символу входного слова, а не ко второму или третьему? В предыдущей задаче мы всегда возвращались к первому из оставшихся символов входного слова, а теперь мы сохраняем все символы, поэтому непонятно, какие символы мы уже скопировали, а какие ещё нет. Отметим также, что в МТ ячейки ленты никак не нумеруются, нет в МТ и счетчиков, которые позволили бы определить, сколько символов мы уже скопировали.

В общем виде проблема, с которой мы столкнулись, следующая: как зафиксировать на ленте некоторую позицию, в которой мы уже были и к которой позже должны вернуться? Обычно эта проблема решается так. Когда мы оказываемся в этой позиции в первый раз, то заменяем находящийся в ней символ на его двойник – на новый вспомогательный символ, причем разные символы заменяем на разные двойники, например a на A и b на B . После этого мы выполняем какие-то действия в других местах ленты. Чтобы затем вернуться к нашей позиции, надо просто отыскать на ленте ту клетку, где находится символ A или B . Затем в данной клетке можно восстановить прежний символ, если нам больше не надо фиксировать эту позицию (именно для восстановления прежнего символа и надо было заменять разные символы на разные двойники).

Воспользуемся этим приёмом в нашей задаче, выполняя следующие действия:

1. Как уже сказано, вначале записываем знак $=$ за входным словом (см. шаг 1 выше).
2. Затем возвращаемся под первый символ входного слова (см. шаг 2 выше).
3. Далее заменяем видимый символ a на двойник A (см. шаг 3 ниже), «бежим» вправо до первой свободной клетки и записываем в неё символ a (см. шаг 4). После этого возвращаемся влево к клетке с двойником A (см.

шаг 5), восстанавливаем прежний символ a и сдвигаемся вправо к следующему символу (см. шаг 6).



Теперь аналогичным образом копируем второй символ (заменяем его на A , в конец дописываем a и т.д.) и все последующие символы входного слова.

4. Когда мы скопируем последний символ входного слова и вернёмся к его двойнику (после шага 12), то затем после сдвига на одну позицию вправо мы попадём на знак $=$ (шаг 13). Это сигнал о том, что входное слово полностью скопировано, поэтому работу МТ надо завершать.

С учётом всего сказанного получаем следующую программу для МТ:

	a	b	=	A	B	Λ	комментарий
q1	,R,	,R,	×	×	×	=,L,q2	поставить = справа от слова
q2	,L,	,L,	×	×	×	,R,q3	налево под 1-й символ
q3	A,R,q4	B,R,q5	,,!	×	×	×	анализ и замена очередного символа
q4	,R,	,R,	,R,	×	×	a,,q6	запись a справа
q5	,R,	,R,	,R,	×	×	b,,q6	запись b справа
q6	,L,	,L,	,L,	a,R,q3	b,R,q3	×	возврат, восстановление, к след. символу

Отметим, что в этой программе можно избавиться от состояния $q6$, если объединить его с состоянием $q2$, предусмотрев в $q2$ возврат влево как до пустой клетки, так и до символов A и B . Выпишем новый вид строки для состояния $q2$:

	a	b	=	A	B	Λ	комментарий
			...				
q2	,L,	,L,	,L,	a,R,q3	b,R,q3	,R,q3	налево до Λ, A или B
			...				

1.3 Задачи для самостоятельного решения

Замечания:

1) В задачах рассматриваются только целые неотрицательные числа, если не сказано иное.

2) Под «единичной» системой счисления понимается запись неотрицательного целого числа с помощью палочек – должно быть выписано столько палочек, какова величина числа; например: $2 \rightarrow ||$, $5 \rightarrow |||||$, $0 \rightarrow$ ⟨пустое слово⟩.

1.1 $A=\{a,b,c\}$. Приписать слева к слову P символ b ($P \rightarrow bP$).

1.2 $A=\{a,b,c\}$. Приписать справа к слову P символы bc ($P \rightarrow Pbc$).

1.3 $A=\{a,b,c\}$. Заменить на a каждый второй символ в слове P .

1.4 $A=\{a,b,c\}$. Оставить в слове P только первый символ (пустое слово не менять).

1.5 $A=\{a,b,c\}$. Оставить в слове P только последний символ (пустое слово не менять).

1.6 $A=\{a,b,c\}$. Определить, является ли P словом ab . Ответ (выходное слово): слово ab , если является, или пустое слово иначе.

1.7 $A=\{a,b,c\}$. Определить, входит ли в слово P символ a . Ответ: слово из одного символа a (да, входит) или пустое слово (нет).

1.8 $A=\{a,b,c\}$. Если в слово P не входит символ a , то заменить в P все символы b на c , иначе в качестве ответа выдать слово из одного символа a .

1.9 $A=\{a,b,0,1\}$. Определить, является ли слово P идентификатором (непустым словом, начинающимся с буквы). Ответ: слово a (да) или пустое слово (нет).

1.10 $A=\{a,b,0,1\}$. Определить, является ли слово P записью числа в двоичной системе счисления (непустым словом, состоящем только из цифр 0 и 1). Ответ: слово 1 (да) или слово 0.

1.11 $A=\{0,1\}$. Считая непустое слово P записью двоичного числа, удалить из него незначащие нули, если такие есть.

1.12 $A=\{0,1\}$. Для непустого слова P определить, является ли оно записью степени двойки $(1, 2, 4, 8, \dots)$ в двоичной системе счисления. Ответ: слово 1 (является) или слово 0.

1.13 $A=\{0,1,2,3\}$. Считая непустое слово P записью числа в четверичной системе счисления, определить, является оно чётным числом или нет. Ответ: 1 (да) или 0.

1.14 $A=\{0,1\}$. Считая непустое слово P записью числа в двоичной системе, получить двоичное число, равное учетверенному числу P (например: $101 \rightarrow 10100$).

1.15 $A=\{0,1\}$. Считая непустое слово P записью числа в двоичной системе, получить двоичное число, равное неполному частному от деления числа P на 2 (например: $1011 \rightarrow 101$).

1.16 $A=\{a,b,c\}$. Если P – слово чётной длины $(0, 2, 4, \dots)$, то выдать ответ a , иначе – пустое слово.

1.17 $A=\{0,1,2\}$. Считая непустое слово P записью числа в троичной системе счисления, определить, является оно чётным числом или нет. Ответ: 1 (да) или 0. (Замечание: в чётном троичном числе должно быть чётное количество цифр 1.)

1.18 $A=\{a,b,c\}$. Пусть P имеет нечётную длину. Оставить в P только средний символ.

1.19 $A=\{a,b,c\}$. Если слово P имеет чётную длину, то оставить в нём только левую половину.

1.20 $A=\{a,b,c\}$. Приписать слева к непустому слову P его первый символ.

1.21 $A=\{a,b\}$. Для непустого слова P определить, входит ли в него ещё раз его первый символ. Ответ: a (да) или пустое слово.

1.22 $A=\{a,b\}$. В непустом слове P поменять местами его первый и последний символы.

1.23 $A=\{a,b\}$. Определить, является P палиндромом (перевёртышем, симметричным словом) или нет. Ответ: a (да) или пустое слово.

1.24 $A=\{a,b\}$. Заменить в P каждое вхождение a на bb .

1.25 $A=\{a,b,c\}$. Заменить в P каждое вхождение ab на c .

1.26 $A=\{a,b\}$. Удвоить слово P (например: $abb \rightarrow abbabb$).

1.27 $A=\{a,b\}$. Удвоить каждый символ слова P (например: $bab \rightarrow bbaabb$).

1.28 $A=\{a,b\}$. Перевернуть слово P (например: $abb \rightarrow bba$).

1.29 $A=\{0,1\}$. Считая непустое слово P записью двоичного числа, получить это же число, но в четверичной системе. (*Замечание:* учесть, что в двоичном числе может быть нечётное количество цифр.)

1.30 $A=\{0,1,2,3\}$. Считая непустое слово P записью числа в четверичной системе счисления, получить запись этого числа в двоичной системе.

1.31 $A=\{0,1,2\}$. Считая непустое слово P записью положительного числа в троичной системе счисления, выполнить действие:

- а) увеличить это число на 1;
- б) уменьшить это число на 1;
- в) умножить это число на 2;
- г) разделить это число на 2 (с отбрасыванием остатка);
- д) найти остаток от деления на 2.

1.32 $A=\{ | \}$. Считая слово P записью числа в единичной системе счисления, получить запись этого числа в троичной системе. (*Рекомендация:* следует в цикле удалять из «единичного» числа по палочке и каждый раз прибавлять 1 к троичному числу, которое вначале положить равным 0.)

1.33 $A=\{0,1,2\}$. Считая непустое слово P записью числа в троичной системе счисления, получить запись этого числа в единичной системе.

1.34 Пусть слово P имеет следующий вид:

$$\underbrace{|\dots|}_n \otimes \underbrace{|\dots|}_m$$

где \otimes – один из знаков $+$, $-$, \times , $/$, \div , \uparrow или \downarrow , слева от которого указано n палочек, а справа – m палочек. Реализовать соответствующую операцию в единичной системе счисления (в качестве ответа выдать слово, указанное справа от стрелки):

а) сложение: $\underbrace{|\dots|}_n + \underbrace{|\dots|}_m \rightarrow \underbrace{|\dots|}_{n+m} \quad (n \geq 0, m \geq 0)$

б) вычитание: $\underbrace{|\dots|}_n - \underbrace{|\dots|}_m \rightarrow \underbrace{|\dots|}_{n-m} \quad (n \geq m \geq 0)$

в) умножение: $\underbrace{|\dots|}_n \times \underbrace{|\dots|}_m \rightarrow \underbrace{|\dots|}_{n \times m} \quad (n \geq 0, m \geq 0)$

г) деление нацело: $\underbrace{||\dots|}_n / \underbrace{||\dots|}_m \rightarrow \underbrace{||\dots|}_k \quad (n \geq 0, m > 0, k = n \operatorname{div} m)$

д) взятие остатка: $\underbrace{||\dots|}_n \div \underbrace{||\dots|}_m \rightarrow \underbrace{||\dots|}_k \quad (n \geq 0, m > 0, k = n \operatorname{mod} m)$

е) максимум: $\underbrace{||\dots|}_n \uparrow \underbrace{||\dots|}_m \rightarrow \underbrace{||\dots|}_k \quad (n \geq 0, m \geq 0, k = \max(n, m))$

ж) минимум: $\underbrace{||\dots|}_n \downarrow \underbrace{||\dots|}_m \rightarrow \underbrace{||\dots|}_k \quad (n \geq 0, m \geq 0, k = \min(n, m))$

1.35 $A = \{ | \}$. Считая слово P записью числа в единичной системе, определить, является ли это число степенью 3 (1, 3, 9, 27, ...). Ответ: пустое слово, если является, или слово из одной палочки иначе.

1.36 $A = \{ | \}$. Считая слово P записью числа n в единичной системе, получить в этой же системе число 2^n .

1.37 $A = \{ | \}$. Пусть слово P является записью числа 2^n ($n=0, 1, 2, \dots$) в единичной системе. Получить в этой же системе число n .

1.38 Пусть P имеет вид $Q+R$, где Q и R – непустые слова из символов 0, 1 и 2. Трактую Q и R как записи чисел в троичной системе счисления (возможно, с незначащими нулями), выдать в качестве ответа запись суммы этих чисел в той же троичной системе.

1.39 Пусть P имеет вид $Q-R$, где Q и R – непустые слова из символов 0, 1 и 2. Трактую Q и R как записи чисел в троичной системе счисления (возможно, с незначащими нулями) и считая, что $Q \geq R$, выдать в качестве ответа запись разности этих чисел в той же троичной системе.

1.40 Пусть P имеет вид $Q=R$, где Q и R – любые слова из символов a и b . Выдать ответ a , если слова Q и R одинаковы, и пустое слово иначе.

1.41 Пусть P имеет вид $Q=R$, где Q и R – непустые слова из символов 0 и 1. Трактую Q и R как записи двоичных чисел (возможно, с незначащими нулями), выдать в качестве ответа слово 1, если эти числа равны, и слово 0 иначе.

1.42 Пусть P имеет вид $Q>R$, где Q и R – непустые слова из символов 0 и 1. Трактую Q и R как записи двоичных чисел (возможно, с незначащими нулями), выдать в качестве ответа слово 1, если число Q больше числа R , и слово 0 иначе.

1.43 $A = \{ (,) \}$. Определить, сбалансировано ли слово P по круглым скобкам. Ответ: D (да) или H (нет).

1.44 $A=\{a,b\}$. Если в P символов a больше, чем символов b , то выдать ответ a , если символов a меньше символов b , то выдать ответ b , а иначе в качестве ответа выдать пустое слово.

2. Нормальные алгоритмы Маркова

В разделе рассматриваются задачи на составление нормальных алгоритмов Маркова. Приводится краткое описание этих алгоритмов, на примерах объясняются основные приёмы их составления и предлагаются задачи для самостоятельного решения.

2.1 Краткое описание нормальных алгоритмов Маркова

Подстановки

Интересной особенностью нормальных алгоритмов Маркова (НАМ) является то, что в них используется лишь одно элементарное действие – так называемая подстановка, которая определяется следующим образом.

Формулой подстановки называется запись вида $\alpha \rightarrow \beta$ (читается « α заменить на β »), где α и β – любые слова (возможно, и пустые). При этом α называется левой частью формулы, а β – правой частью.

Сама **подстановка** (как действие) задается формулой подстановки и применяется к некоторому слову P . Суть операции сводится к тому, что в слове P отыскивается часть, совпадающая с левой частью этой формулы (т.е. с α), и она заменяется на правую часть формулы (т.е. на β). При этом остальные части слова P (слева и справа от α) не меняются. Получившееся слово R называют **результатом подстановки**. Условно это можно изобразить так:

$$P \begin{array}{|c|c|c|} \hline x & \alpha & y \\ \hline \end{array} \rightarrow R \begin{array}{|c|c|c|} \hline x & \beta & y \\ \hline \end{array}$$

Необходимые уточнения:

1. Если левая часть формулы подстановки входит в слово P , то говорят, что эта формула **применима к P** . Но если α не входит в P , то формула считается **неприменимой к P** , и подстановка не выполняется.

2. Если левая часть α входит в P несколько раз, то на правую часть β , по определению, заменяется только первое (самое левое) вхождение α в P :

$$P \begin{array}{|c|c|c|c|c|} \hline x & \alpha & y & \alpha & z \\ \hline \end{array} \rightarrow R \begin{array}{|c|c|c|c|c|} \hline x & \beta & y & \alpha & z \\ \hline \end{array}$$

3. Если правая часть формулы подстановки – пустое слово (в таком слове нет ни одного символа), то подстановка $\alpha \rightarrow$ сводится к вычеркиванию части α из P (отметим попутно, что в формулах подстановки не принято как-либо обозначать пустое слово):

$$P \begin{array}{|c|c|c|} \hline x & \alpha & y \\ \hline \end{array} \rightarrow R \begin{array}{|c|c|} \hline x & y \\ \hline \end{array}$$

4. Если в левой части формулы подстановки указано пустое слово, то подстановка $\rightarrow \beta$ сводится, по определению, к приписыванию β слева к слову P :

$$P \begin{array}{|c|} \hline x \\ \hline \end{array} \rightarrow R \begin{array}{|c|c|} \hline \beta & x \\ \hline \end{array}$$

Из этого правила вытекает очень важный факт: формула с пустой левой частью применима к любому слову. (Пустое слово всегда присутствует перед любым словом). Отметим также, что формула с пустыми левой и правой частями не меняет слово.

Определение НАМ

Нормальным алгоритмом Маркова (НАМ) называется непустой конечный упорядоченный набор формул подстановки:

$$\begin{cases} \alpha_1 \rightarrow \beta_1 \\ \alpha_2 \rightarrow \beta_2 \\ \dots \\ \alpha_k \rightarrow \beta_k \end{cases} \quad (k \geq 1)$$

В этих формулах могут использоваться два вида стрелок: обычная стрелка (\rightarrow) и стрелка «с хвостиком» (\mapsto). Формула с обычной стрелкой называется **обычной формулой**, а формула со стрелкой «с хвостиком» – **заключительной формулой**. Разница между ними объясняется чуть ниже.

Записать алгоритм в виде НАМ – значит предъявить такой набор формул.

Правила выполнения НАМ

К началу выполнения НАМ задается **входное слово** P . Где именно и кем оно записано – не важно, в НАМ этот вопрос не оговаривается.

Работа НАМ сводится к выполнению последовательности шагов. На каждом шаге входящие в НАМ формулы подстановки просматриваются сверху вниз и выбирается первая из формул, применимых к входному слову P , т.е. самая верхняя из тех, левая часть которых входит в P . Далее выполняется подстановка согласно найденной формуле. Получается новое слово P' .

На следующем шаге это слово P' берется за исходное и к нему применяется та же самая процедура, т.е. формулы снова просматриваются сверху вниз начиная с самой верхней и ищется первая формула, применимая к слову P' , после чего выполняется соответствующая подстановка и получается новое слово P'' . И так далее:

$$P \rightarrow P' \rightarrow P'' \rightarrow \dots$$

Следует обратить особое внимание на тот факт, что на каждом шаге формулы в НАМ всегда просматриваются начиная с самой первой.

Необходимые уточнения:

1. Если на очередном шаге была применена обычная формула ($\alpha \rightarrow \beta$), то работа НАМ продолжается (переход к следующему шагу).

2. Если же на очередном шаге была применена заключительная формула ($\alpha \mapsto \beta$), то после её применения работа НАМ прекращается. То слово, которое получилось в этот момент, и есть **выходное слово**, т.е. результат применения НАМ к входному слову.

Как видно, разница между обычной и заключительной формулами подстановки проявляется лишь в том, что после применения обычной формулы работа НАМ продолжается, а после заключительной формулы – прекращается.

3. Если на очередном шаге к текущему слову неприменима ни одна формула, то и в этом случае работа НАМ прекращается, а **выходным словом** считается текущее слово.

Таким образом, НАМ останавливается по двум причинам: либо была применена заключительная формула, либо ни одна из формул не

подошла. То и другое считается «хорошим» окончанием работы НАМ. В обоих случаях говорят, что НАМ *применím* к входному слову.

Однако может случиться и так, что НАМ никогда не остановится; это происходит, когда на каждом шаге есть применимая формула и эта формула обычная. Тогда говорят, что НАМ *неприменím* к входному слову. В этом случае ни о каком результате нет и речи.

2.2 Примеры на составление НАМ

Рассмотрим примеры, в которых демонстрируются типичные приёмы составления НАМ.

Как и в случае машины Тьюринга, для сокращения формулировки задач будем использовать следующие соглашения:

- буквой P будем обозначать входное слово;
- буквой A будем обозначать алфавит входного слова, т.е. набор тех символов, которые и только которые могут входить во входное слово P (но в процессе выполнения НАМ в обрабатываемых словах могут появляться и другие символы).

Кроме того, в примерах будем справа от формул подстановки указывать их номера. Эти номера не входят в формулы, а нужны для ссылок на формулы при показе пошагового выполнения НАМ.

Пример 1 (удаление и вставка символов)

$A = \{a, b, c, d\}$. В слове P требуется удалить все вхождения символа c , а затем заменить первое вхождение подслова bb на ddd .

Например: $abbcabbca \rightarrow adddabba$

Решение

Прежде всего отметим, что в НАМ, в отличие от машины Тьюринга, легко реализуются вставки и удаления символов. Вставка новых символов в слово – это замена некоторого подслова на подслово с бóльшим числом символов; например, с помощью формулы $bb \rightarrow ddd$ два символа будут заменены на три символа. При этом не надо заботиться о том, чтобы предварительно освободить место для дополнительных символов, в НАМ слово раздвигается автоматически. Удаление же символов – это замена некоторого подслова на подслово с

меньшим числом символов; например, удаление символа c реализуется формулой $c \rightarrow$ (c пустой правой частью). При этом никаких пустых позиций внутри слова не появляется, сжатие слова в НАМ происходит автоматически.

С учётом сказанного нашу задачу должен, казалось бы, решать такой НАМ:

$$\begin{cases} c \rightarrow & (1) \\ bb \rightarrow ddd & (2) \end{cases}$$

Однако это не так. Проверим этот НАМ на входном слове $abbcbabbca$ (над стрелками указаны номера применённых формул, а в словах слева от стрелок подчёркнуты для наглядности те части, к которым были применены эти формулы):

$$abb\underset{1}{c}abbca \rightarrow abbabb\underset{1}{c}a \rightarrow \underset{2}{ab}b\underset{2}{ab}ba \rightarrow addd\underset{2}{ab}ba \rightarrow$$

$\rightarrow adddaddda$

Как видно, НАМ сначала удалил все символы c и только затем заменил первое вхождение bb на ddd . Однако НАМ на этом не остановился и стал заменять остальные вхождения bb . Почему? Дело в том, что, пока применима хотя бы одна формула, НАМ продолжает свою работу. Но нам этого не надо, поэтому мы должны принудительно остановить НАМ после того, как он заменил первое вхождение bb . Вот для этого и нужны заключительные формулы подстановки, после применения которых НАМ останавливается. Следовательно, в нашем алгоритме обычную формулу $bb \rightarrow ddd$ надо заменить на заключительную формулу $bb \mapsto ddd$:

$$\begin{cases} c \rightarrow & (1) \\ bb \mapsto ddd & (2) \end{cases}$$

Вот теперь наш алгоритм будет работать правильно:

$$abb\underset{1}{c}abbca \rightarrow abbabb\underset{1}{c}a \rightarrow \underset{2}{ab}b\underset{2}{ab}ba \mapsto adddabba$$

Слово, которое получилось после применения заключительной формулы (2), является выходным словом, т.е. результатом применения НАМ к заданному входному слову.

Проверим наш НАМ ещё и на входном слове, в которое не входит bb :

$$d\underset{1}{c}ac\underset{1}{b} \rightarrow d\underset{1}{a}c\underset{1}{b} \rightarrow dab$$

К последнему слову (dab) неприменима ни одна формула, поэтому, согласно определению НАМ, алгоритм останавливается и это слово объявляется выходным.

Пример 2 (перестановка символов)

$A = \{a, b\}$. Преобразовать слово P так, чтобы в его начале оказались все символы a , а в конце – все символы b .

Например: $babba \rightarrow aabbb$

Решение

Казалось бы, для решения этой задачи нужен сложный НАМ. Однако это не так, задача решается с помощью НАМ, содержащего всего одну формулу:

$$\{ba \rightarrow ab\}$$

Пока в слове P справа хотя бы от одного символа b есть символ a , эта формула будет переносить a налево от этого b . Формула перестает работать, когда справа от b нет ни одного a , это и означает, что все a оказались слева от b . Например:

$$\underline{b}abba \rightarrow ab\underline{b}ba \rightarrow ab\underline{b}ab \rightarrow ab\underline{a}bb \rightarrow aabbb$$

Алгоритм остановился на последнем слове, т.к. к нему уже неприменима наша формула.

Этот и предыдущий примеры показывают, что в НАМ, в отличие от машины Тьюринга, легко реализуются перестановки, вставки и удаления символов. Однако в НАМ возникает другая проблема: как зафиксировать символ (подслово), который должен быть обработан? Рассмотрим эту проблему на следующем примере.

Пример 3 (использование спецзнака)

$A = \{a, b\}$. Удалить из непустого слова P его первый символ. Пустое слово не менять.

Решение

Ясно, что удалив первый символ слова, надо тут же остановиться. Поэтому, казалось бы, задачу решает следующий НАМ:

$$\begin{cases} a \mapsto & (1) \\ b \mapsto & (2) \end{cases}$$

Однако это неправильный алгоритм, в чём можно убедиться, применив его к слову *bbaba*:

$$\overset{1}{\text{bbaba}} \mapsto \text{bbba}$$

Как видно, этот НАМ удалил не первый символ слова, а первое вхождение символа *a*, а это разные вещи. Данный алгоритм будет правильно работать, только если входное слово начинается с символа *a*. Ясно, что перестановка формул в этом НАМ не поможет, т.к. тогда он будет, напротив, неправильно работать на словах, начинающихся с *a*.

Что делать? Надо как-то зафиксировать, пометить первый символ слова, например, поставив перед ним какой-либо знак, скажем *, отличный от символов алфавита слова. После этого уже можно с помощью формул вида $*\xi \mapsto$ заменить этот знак и первый символ ξ слова на пусто и остановиться:

$$\text{bbaba} \rightarrow * \text{bbaba} \mapsto \text{baba}$$

А как поставить * перед первым символом? Это реализуется формулой $\rightarrow*$ с пустой левой частью, которая, по определению, приписывает свою правую часть слева к слову.

Итого, получаем следующий НАМ:

$$\left\{ \begin{array}{l} \rightarrow * \quad (1) \\ * a \mapsto \quad (2) \\ * b \mapsto \quad (3) \end{array} \right.$$

Проверим его на том же входном слове:

$$\overset{1}{\text{bbaba}} \rightarrow \overset{1}{*} \overset{1}{\text{bbaba}} \rightarrow \overset{1}{**} \overset{1}{\text{bbaba}} \rightarrow \dots$$

Как видно, этот алгоритм постоянно приписывает слева звёздочки. Почему? Напомним, что формула подстановки с пустой левой частью применима всегда, поэтому наша формула (1) будет работать бесконечно, блокируя доступ к остальным формулам. Отсюда вытекает очень важное правило: если в НАМ есть формула с пустой левой частью ($\rightarrow\beta$), то её место – только в самом конце НАМ. Учтём это правило и перепишем наш НАМ:

$$\left\{ \begin{array}{l} * a \mapsto \quad (1) \\ * b \mapsto \quad (2) \\ \rightarrow * \quad (3) \end{array} \right.$$

Проверим данный алгоритм:

$$\text{bbaba} \rightarrow \overset{3}{*} \overset{2}{\text{bbaba}} \mapsto \text{baba}$$

Казалось бы, всё в порядке. Однако это не так: наш алгоритм заикнется на пустом входном слове, т.к. постоянно будет применяться формула (3), а согласно условию задачи на таком слове НАМ должен остановиться. В чём причина этой ошибки? Дело в том, что мы ввели знак * для того, чтобы пометить первый символ слова, а затем уничтожить * и этот символ. Но в пустом слове нет ни одного символа, поэтому формулы (1) и (2) ни разу не сработают и постоянно будет выполняться формула (3). Следовательно, чтобы учесть случай пустого входного слова, надо после формул (1) и (2) записать ещё одну формулу, которая уничтожает «одинокую» звёздочку и останавливает алгоритм:

$$\left\{ \begin{array}{ll} *a \mapsto & (1) \\ *b \mapsto & (2) \\ * \mapsto & (3) \\ \rightarrow * & (4) \end{array} \right.$$

Вот теперь мы, наконец-то, составили правильный алгоритм.

Прежде чем перейти к следующим задачам, обобщим тот приём со звёздочкой, который мы использовали в примере 3.

Пусть в обрабатываемое слово P входит несколько раз подслово α :

$$P \quad \boxed{\dots \mid \alpha \mid \dots \mid \alpha \mid \dots \mid \alpha \mid \dots}$$

и нам надо заменить одно из вхождений α на подслово β . Такая замена делается с помощью формулы $\alpha \rightarrow \beta$. Однако, если мы применим эту формулу к слову P , то будет заменено первое вхождение α . А что делать, если надо заменить какое-то другое вхождение α , скажем второе или последнее? Так вот, чтобы на β заменялось не первое вхождение α , а какое-то другое, это другое вхождение надо как-то выделить, пометить, для чего следует рядом с ним (слева или справа) поставить некоторый символ, скажем *, отличный от всех других символов, входящих в P :

$$P \quad \boxed{\dots \mid \alpha \mid \dots \mid * \alpha \mid \dots \mid \alpha \mid \dots}$$

Такой символ будем в дальнейшем называть *спецзнаком* (аналог вспомогательного символа в МТ). Его роль – выделить нужное вхождение α среди других, сделать его уникальным. Поскольку только около этого вхождения есть спецзнак, то надо использовать

формулу $*\alpha \rightarrow \beta$, чтобы заменить на β именно это вхождение α , а не какое-то другое.

Этот приём со спецзнаком следует запомнить, т.к. в НАМ он используется очень часто.

Правда, остаётся ещё одна проблема: как спецзнак разместить рядом с нужным вхождением α ? Следующие примеры показывают, как это делается.

Пример 4 (фиксация спецзнаком обрабатываемого символа)

$A = \{0,1,2,3\}$. Пусть P – непустое слово. Трактую его как запись неотрицательного целого числа в четверичной системе счисления, требуется получить запись этого же числа, но в двоичной системе.

Например: $0123 \rightarrow 00011011$

Решение

Как известно, для перевода числа из четверичной системы в двоичную надо каждую четверичную цифру заменить на пару соответствующих ей двоичных цифр: $0 \rightarrow 00$, $1 \rightarrow 01$, $2 \rightarrow 10$, $3 \rightarrow 11$. Такая замена, казалось бы, реализуется следующим НАМ:

$$\left\{ \begin{array}{ll} 0 \rightarrow 00 & (1) \\ 1 \rightarrow 01 & (2) \\ 2 \rightarrow 10 & (3) \\ 3 \rightarrow 11 & (4) \end{array} \right.$$

Но этот алгоритм неправильный, в чём можно убедиться на входном слове 0123 :

$$\overset{1}{0}123 \rightarrow \overset{1}{00}123 \rightarrow \overset{1}{000}123 \rightarrow \dots$$

Ошибка здесь в том, что после замены четверичной цифры на пару двоичных цифр уже никак нельзя отличить двоичные цифры от четверичных, поэтому наш НАМ начинает заменять и двоичные цифры. Значит, надо как-то отделить ту часть числа, в которой уже была произведена замена, от той части, где замены ещё не было. Для этого предлагается пометить слева спецзнаком $*$ ту четверичную цифру, которая сейчас должна быть заменена на пару соответствующих двоичных цифр, а после того как такая замена будет выполнена, спецзнак нужно поместить перед следующей четверичной цифрой:

$$0123 \rightarrow *0123 \rightarrow 00*123 \rightarrow 0001*23 \rightarrow 000110*3 \rightarrow$$

→ 00011011*

Как видно, слева от спецзнака всегда находится та часть числа, которая уже переведена в двоичный вид, а справа – часть, которую ещё предстоит заменить. Поэтому никакой путаницы между четверичными и двоичными цифрами уже не будет.

Итак, спецзнак * сначала должен быть размещён слева от первой цифры четверичного числа, а затем он должен «перепрыгивать» через очередную четверичную цифру, оставляя слева от себя соответствующие ей двоичные цифры. В конце же, когда справа от * уже не окажется никакой цифры, спецзнак надо уничтожить и остановиться. Как приписать * слева к входному слову и как уничтожить спецзнак с остановом, мы уже знаем по предыдущему примеру, а вот «перепрыгивание» звёздочки реализуется с помощью формул вида $*\alpha \rightarrow \beta\gamma*$, где α – четверичная цифра, а $\beta\gamma$ – соответствующая ей пара двоичных цифр.

Итого, получаем следующий алгоритм перевода чисел из четверичной системы в двоичную:

$$\left\{ \begin{array}{ll} *0 \rightarrow 00* & (1) \\ *1 \rightarrow 01* & (2) \\ *2 \rightarrow 10* & (3) \\ *3 \rightarrow 11* & (4) \\ * \mapsto & (5) \\ \rightarrow * & (6) \end{array} \right.$$

Проверим этот НАМ на входном слове 0123:

0123 → $\overset{6}{*}0\overset{1}{1}2\overset{2}{3} \rightarrow 00\overset{1}{*}1\overset{2}{2}3 \rightarrow 0001\overset{3}{*}2\overset{4}{3} \rightarrow 000110\overset{4}{*}3 \rightarrow 00011011\overset{5}{*} \mapsto$

$\mapsto 00011011$

Пример 5 (перемещение спецзнака)

$A = \{a, b\}$. Требуется приписать символ a к концу слова P .

Например: $bbab \rightarrow bbaba$

Решение

Прежде всего напомним, что формула $\rightarrow a$ приписывает символ a слева к слову P , а не справа. Чтобы приписать a справа, надо сначала

пометить конец слова. Для этого воспользуемся спецзнаком, который поместим в конец P , а затем заменим его на a :

$$P \rightarrow \dots \rightarrow P^* \mapsto Pa$$

Но как поместить спецзнак в конец слова? Делается это так: сначала спецзнак $*$ приписываем слева к слову P , а затем «перегоняем» звёздочку через все буквы слова:

$$bbab \rightarrow *bbab \rightarrow b*bab \rightarrow bb*ab \rightarrow bba*b \rightarrow bbab^*$$

А как сделать такой перегон? Нетрудно заметить, что «перепрыгивание» звёздочки через какой-то символ ξ – это замена пары $*\xi$ на пару ξ^* , которая реализуется формулой $*\xi \rightarrow \xi^*$.

С учётом всего сказанного получаем следующий НАМ:

$$\left\{ \begin{array}{l} *a \rightarrow a^* \\ *b \rightarrow b^* \\ * \mapsto a \\ \rightarrow * \end{array} \right.$$

Отметим, что при пустом входном слове этот НАМ сначала введёт звёздочку, а затем тут же заменит её на символ a и остановится.

Пример 6 (смена спецзнака)

$A = \{a, b\}$. В слове P заменить на aa последнее вхождение символа a , если такое есть. Например: $bababb \rightarrow babaabb$

Решение

Удвоение символа a реализуется формулой $a \mapsto aa$. Но чтобы она применялась не к первому вхождению символа a , а к последнему, надо поставить, скажем, справа от последнего символа a спецзнак $*$ и применить формулу $a^* \mapsto aa$.

Теперь посмотрим, как поместить $*$ рядом с последним вхождением символа a . Поскольку последнее вхождение – это первое вхождение от конца, то предлагается сначала приписать $*$ слева к слову P , затем перегнуть $*$ в конец слова (это мы уже умеем делать), а далее перегнуть $*$ справа налево через символы b до ближайшего символа a . Кроме того, надо учесть, что в P может и не быть символа a ; поэтому, если

звёздочка снова достигнет начала слова, её надо уничтожить и остановиться.

Реализуем эту идею в виде следующего НАМ:

$$\left\{ \begin{array}{l} *a \rightarrow a* \quad (1) \\ *b \rightarrow b* \quad (2) \\ b* \rightarrow *b \quad (3) \\ a* \mapsto aa \quad (4) \\ * \mapsto \quad (5) \\ \rightarrow * \quad (6) \end{array} \right.$$

Здесь формула (6) приписывает * слева к входному слову P , формулы (1) и (2) перегоняют * в конец P , после чего формула (3) перемещает * справа налево через все b в конце слова до ближайшего, т.е. последнего, символа a , и, наконец, формула (4) заменяет этот символ на aa и останавливает алгоритм. Формула же (5) нужна для входных слов, в которые не входит a .

Проверим этот алгоритм на входном слове $bababb$ (двойная стрелка означает несколько шагов применения формул (1) и (2)):

$$\begin{array}{ccccccc} & 6 & & 1,2 & & 3 & & 2 & & 3 \\ bababb & \rightarrow & *bababb & \Rightarrow & bababb* & \rightarrow & babab*b & \rightarrow & bababb* & \rightarrow \\ & & & & & & & & & & 2 \\ \rightarrow & babab*b & \rightarrow & \dots & & & & & & & \end{array}$$

Как видно, вместо того, чтобы двигаться справа влево до ближайшего символа a , звёздочка начала «прыгать» вокруг последнего символа слова. Почему? Дело в том, что формулы (1) и (2), перегоняющие * вправо, мешают формуле (3), перегоняющей * влево. Отметим, что перестановка этих формул не поможет, т.к. тогда * начнёт «плясать» вокруг первого символа входного слова. Что делать?

Ошибка произошла из-за того, что мы используем спецзнак * в двух разных целях – как для движения вправо, так и для движения влево. Так вот, чтобы этой ошибки не было, надо просто ввести еще один спецзнак, скажем #, распределив между этими спецзнаками обязанности: пусть * движется вправо, а # – влево. Появиться же спецзнак # должен тогда, когда * дойдет до конца слова, т.е. когда справа от * не окажется других символов. Такая замена спецзнака «исключит из игры» все формулы со звёздочкой в левой части, поэтому они уже не будут мешать формулам со спецзнаком #.

Если так и сделать, то получим следующий НАМ:

$$\left\{ \begin{array}{l} *a \rightarrow a* \quad (1) \\ *b \rightarrow b* \quad (2) \\ * \rightarrow \# \quad (3) \\ b\# \rightarrow \#b \quad (4) \\ a\# \mapsto aa \quad (5) \\ \# \mapsto \quad (6) \\ \rightarrow * \quad (7) \end{array} \right.$$

Проверим этот алгоритм на прежнем входном слове:

$$\begin{array}{cccccccc} & 7 & & 1,2 & & 3 & & 4 & & 4 \\ bababb & \rightarrow & *bababb & \Rightarrow & bababb* & \rightarrow & bababb\# & \rightarrow & babab\#b & \rightarrow \\ & & & & & & & & & 5 \\ baba\#bb & \mapsto & babaabb & & & & & & & \end{array}$$

Если же во входное слово не входит символ a , тогда имеем:

$$\begin{array}{cccccccc} & 7 & & 2 & & 2 & & 3 & & 4 & & 4 & & 6 \\ bb & \rightarrow & *bb & \rightarrow & b*b & \rightarrow & bb* & \rightarrow & bb\# & \rightarrow & b\#b & \rightarrow & \#bb & \mapsto & bb \end{array}$$

Пример 7 (перенос символа через слово)

$A=\{a,b\}$. Перенести в конец непустого слова P его первый символ. Пустое слово не менять.

Например: $bbaba \rightarrow babab$

Решение

Для решения этой задачи предлагается выполнить следующие действия.

1. Помечаем первый символ слова P спецзнаком $*$.
2. Заменяем $*$ и этот символ на новый символ: a на A , b на B . Этим мы фактически вводим два новых спецзнака A и B , которые нужны, чтобы отличить первый символ слова от остальных символов при его переносе в конец слова.
3. Перегоняем новый символ A или B через все символы слова P в его конец. Такое перемещение реализуется аналогично перегону звёздочки – с помощью формул вида $A\xi \rightarrow \xi A$ и $B\xi \rightarrow \xi B$
4. Наконец, заменяем A или B в конце слова на прежний символ (A на a , B на b) и останавливаем алгоритм.

Все эти действия реализуются в виде следующего НАМ:

$$\left\{ \begin{array}{ll}
 *a \rightarrow A & (1) \\
 *b \rightarrow B & (2) \\
 Aa \rightarrow aA & (3) \\
 Ab \rightarrow bA & (4) \\
 Ba \rightarrow aB & (5) \\
 Bb \rightarrow bB & (6) \\
 A \mapsto a & (7) \\
 B \mapsto b & (8) \\
 * \mapsto & (9) \\
 \rightarrow * & (10)
 \end{array} \right.$$

Здесь формулы (1) и (2) заменяют первый символ слова (вместе с *) на A или B . Формулы (3)–(6) перегоняют A и B в конец слова. Формулы (7) и (8) применяются только тогда, когда A и B окажутся в конце слова (когда за ними уже нет символов), и восстанавливают исходный вид первого символа. Формула (9) нужна на случай пустого входного слова. Формула же (10) ставит спецзнак * перед первым символом.

Проверим этот алгоритм на входном слове $bbaba$ и на входном слове из одного символа:

$$\begin{array}{cccccccc}
 & 10 & & 2 & & 6 & & 5 & & 6 & & 5 \\
 bbaba & \rightarrow & \underline{*}bbaba & \rightarrow & \underline{B}baba & \rightarrow & b\underline{B}aba & \rightarrow & ba\underline{B}ba & \rightarrow & bab\underline{B}a & \rightarrow \\
 & & & & & & & & & & & 8 \\
 & & & & & & & & & & & \mapsto & babab \\
 & 10 & & 1 & & 7 & & & & & & & \\
 a & \rightarrow & \underline{*}a & \rightarrow & \underline{A} & \mapsto & a & & & & & &
 \end{array}$$

Другое решение

Отметим, что в этом НАМ можно уменьшить число формул, если не вводить новые символы A и B , а использовать вместо них пары $*a$ и $*b$. Это позволит исключить из НАМ формулы (1) и (2), которые вводят символы A и B , и формулы (7) и (8), которые восстанавливают первый символ. Что же касается «перепрыгивания» этих пар через какой-то символ ξ , то оно реализуется формулами вида $*a\xi \rightarrow \xi*a$ и $*b\xi \rightarrow \xi*b$. При этом никакой путаницы между первым символом слова и остальными символами не будет, т.к. только перед первым символом находится звёздочка.

Итак, возможен и следующий НАМ, решающий нашу задачу:

и с символом =), перенося тем самым A в конец слова, где и заменяем на a :

$$abb= \rightarrow aAbb= \rightarrow abAb= \rightarrow abbA= \rightarrow abb=A \rightarrow abb=a$$

Аналогично копируются и символы b .

Главный вопрос здесь: как узнать, какой именно символ исходного слова мы только что скопировали и какой символ надо копировать следующим? Для этого используем стандартный приём со спецзнаком – будем помечать новым спецзнаком $\#$ тот символ, который должен копироваться следующим (вначале это первый символ входного слова):

$$\#abb= \rightarrow a\#Abb= \rightarrow a\#bAb= \rightarrow a\#bbA= \rightarrow a\#bb=A \rightarrow a\#bb=a$$

Как только копия очередного символа окажется в конце, спецзнак $\#$ должен «запустить» процесс копирования следующего символа:

$$\begin{aligned} a\#bb=a &\rightarrow ab\#Bb=a \rightarrow ab\#bB=a \rightarrow ab\#b=Va \rightarrow ab\#b=aB \rightarrow \\ &\rightarrow ab\#b=ab \rightarrow abb\#B=ab \rightarrow abb\#=Bab \rightarrow abb\#=aBb \rightarrow \\ &\rightarrow abb\#=abB \rightarrow abb\#=abb \end{aligned}$$

Когда справа от спецзнака $\#$ окажется символ =, это будет означать, что входное слово полностью скопировано. Осталось только уничтожить символы $\#$ и =, после чего остановиться.

Теперь отметим, что в НАМ, реализующем такое копирование, важен взаимный порядок расположения формул ($A\xi \rightarrow \xi A$, $B\xi \rightarrow \xi B$, $A \rightarrow a$ и $B \rightarrow b$), которые переносят символы A и B в конец и там восстанавливают символы a и b , и формулами ($\#a \rightarrow a\#A$ и $\#b \rightarrow b\#B$), которые «вводят в игру» символы A и B . Поскольку последняя пара формул должна срабатывать только после того, как символ A или B будет полностью перенесён в конец и заменён на a или b , то эта пара формулы должна располагаться в НАМ ниже всех первых формул.

И ещё один момент. В этом НАМ используются два спецзнака $*$ и $\#$, первый из которых нужен для приписывания символа = справа к входному слову, а второй – для указания, какой символ слова должен копироваться следующим. Как ввести эти спецзнаки? Отметим, что использовать для этого две формулы \rightarrow^* и $\rightarrow\#$ нельзя, т.к. первая из них будет блокировать доступ ко второй. Оба этих спецзнака надо вводить сразу одной формулой $\rightarrow\#^*$. При этом надо учитывать, что формулы с $*$ должны применяться самыми первыми, поэтому они должны располагаться в начале НАМ. Формулы же с $\#$, A и B должны распо-

лагаться ниже, чтобы они работали только после того, как исчезнет * и появится символ =.

С учётом всего сказанного получаем следующий НАМ:

$$\left\{ \begin{array}{ll} *a \rightarrow a* & (1) \\ *b \rightarrow b* & (2) \\ * \rightarrow = & (3) \\ Aa \rightarrow aA & (4) \\ Ab \rightarrow bA & (5) \\ A= \rightarrow =A & (6) \\ A \rightarrow a & (7) \\ Ba \rightarrow aB & (8) \\ Bb \rightarrow bB & (9) \\ B= \rightarrow =B & (10) \\ B \rightarrow b & (11) \\ \#a \rightarrow a\#A & (12) \\ \#b \rightarrow b\#B & (13) \\ \#= \mapsto & (14) \\ \rightarrow \#* & (15) \end{array} \right.$$

Здесь формулы (1)–(3) «перегоняют» звёздочку в конец входного слова и заменяют её на символ =. Формулы (4)–(7) перегоняют символ A в конец слова, после чего заменяют на a . Формулы (8)–(11) делают то же самое с B и b . Формулы (12) и (13) «вводят в игру» символы A и B , соответствующие тому символу входного слова, который должен быть скопирован следующим. Формула (14) применяется только тогда, когда справа от $\#$ нет ни a , ни b , т.е. когда полностью просмотрено всё входное слово. И, наконец, формула (15) вводит сразу два спецзнака $\#$ и $*$.

Проверим данный алгоритм на двух входных словах – на пустом и на abb :

$$\begin{aligned} \langle \text{пустое слово} \rangle &\xrightarrow{15} \#* \xrightarrow{3} \#= \mapsto (\text{т.е. получили} \\ &\qquad\qquad\qquad \langle \text{пустое слово} \rangle \langle \text{пустое слово} \rangle) \\ \\ abb &\xrightarrow{15} \#*abb \xrightarrow{1,2} \#abb* \xrightarrow{3} \#abb= \xrightarrow{12} a\#Abb= \xrightarrow{4-6} a\#bb=\underline{A} \xrightarrow{7} \\ &\xrightarrow{13} a\#bb=a \xrightarrow{8-10} ab\#Bb= \xrightarrow{11} ab\#b=\underline{a}B \xrightarrow{13} ab\#b=ab \xrightarrow{8-10} abb\#B=ab \Rightarrow \end{aligned}$$

$$\Rightarrow abb\# = ab\underline{b} \rightarrow abb\# = abb \mapsto abbabb$$

Другое решение

Приведём ещё одно решение задачи удвоения слова, где предлагается выполнить следующие действия.

1. Сначала за каждой (малой) буквой входного слова вставляем её двойник – соответствующую большую букву. Для этого приписываем слева к слову спецзнак *, а затем переносим его через каждую малую букву так, чтобы слева от него остались эта буква и её двойник. В конце слова заменяем * на новый спецзнак #, который нам ещё понадобится:

$$abb \rightarrow *abb \rightarrow aA*bb \rightarrow aAbV*b \rightarrow aAbVbV* \rightarrow aAbVbV\#$$

2. В получившемся слове переставляем малые и большие буквы так, чтобы слева оказались все малые буквы, а справа – все большие, сохраняя при этом исходный взаимный порядок как среди малых, так и среди больших букв:

$$aAbVbV\# \rightarrow \dots \rightarrow abbABV\#$$

3. Как видно, справа мы получили копию входного слова, но записанную большими буквами. Осталось только заменить их на малые буквы. Вот здесь-то и понадобится спецзнак #: будем перемещать его влево через каждую большую букву с заменой её на малую. Когда слева от # уже не окажется большой буквы, надо уничтожить # и остановиться:

$$abbABV\# \rightarrow abbAB\#b \rightarrow abbA\#bb \rightarrow abb\#abb \mapsto abbabb$$

Все указанные действия описываются в виде следующего НАМ:

$$\left\{ \begin{array}{l} *a \rightarrow aA* \\ *b \rightarrow bB* \\ * \rightarrow \# \\ Aa \rightarrow aA \\ Ab \rightarrow bA \\ Va \rightarrow aV \\ Vb \rightarrow bV \\ A\# \rightarrow \#a \\ B\# \rightarrow \#b \\ \# \mapsto \\ \rightarrow * \end{array} \right.$$

Пример 9 (согласованная работа с различными частями слова)
 $A = \{a, b\}$. Пусть слово P имеет чётную длину $(0, 2, 4, \dots)$. Удалить правую
половину этого слова. Например: $bbab \rightarrow bb$

Решение

За основу решения задачи возьмём следующую идею: согласуем просмотр очередной буквы из левой половины с удалением буквы из правой половины. Для достижения этой цели предлагается следующий план действий.

1. Приписываем спецзнак $*$ (звездочку) слева к входному слову P .

2. Звездочка сигнализирует, что необходимо обработать следующую за ней букву (если она есть) или прекратить обработку слова, если справа от звездочки ничего нет. Обработка буквы состоит в следующем. Звездочка перепрыгивает через букву и одновременно выпускает «стрелу»: спецзнак $|$ появляется справа от звездочки. Далее стрела «летит» через буквы к концу слова и «поражает» (т.е. удаляет) последнюю букву. Сама «стрела» при этом тоже исчезает.

3. Действия пункта 2 продолжаются до тех пор пока звездочка не окажется справа от текущего слова, что будет сигнализировать об окончании процесса удаления правой половины слова. После этого сигнала остается удалить спецзнак $*$ и закончить выполнение алгоритма.

Эти действия реализуются с помощью следующего НАМ:

$$\left\{ \begin{array}{ll} |a \rightarrow a| & (1) \\ |b \rightarrow b| & (2) \\ a| \rightarrow & (3) \\ b| \rightarrow & (4) \\ *a \rightarrow a*| & (5) \\ *b \rightarrow b*| & (6) \\ * \mapsto & (7) \\ \rightarrow * & (8) \end{array} \right.$$

Формула (8) приписывает звездочку к началу слова, и она становится своеобразной меткой текущей буквы, всегда находящейся справа от $*$.

Формулы (5) и (6) переносят звездочку через текущую букву и порождают справа от нее «стрелу» (спецзнак $|$). Формулы (1) и (2)

перегоняют | в конец слова, а формулы (3) и (4) удаляют этот спецзнак вместе с последней буквой (тем самым наш спецзнак | на время исчезает, что позволяет звездочке вновь начать работу). Далее снова выполняются формулы (5) и (6), то есть происходит переход к аналогичной обработке очередного символа из левой половины, помеченного звездочкой. Конец этого циклического процесса – достижение звёздочкой правого края текущего слова, после чего выполняется формула (7), приводящая к останову.

Продemonстрируем работу НАМ на слове $P = bbab$:

$$\begin{array}{cccccccc}
 & 8 & & 6 & & 2 & & 1 & & 2 & & 4 \\
 bbab & \rightarrow & *bbab & \rightarrow & b*|bab & \rightarrow & b*b|ab & \rightarrow & b*ba|b & \rightarrow & b*bab| & \rightarrow \\
 & & 6 & & 1 & & 3 & & 7 & & & \\
 \rightarrow & b*ba & \rightarrow & bb*|a & \rightarrow & bb*a| & \rightarrow & bb* & \mapsto & bb & &
 \end{array}$$

Другое решение

Приведём ещё одно, весьма оригинальное, решение этой задачи. Оно предполагает последовательное выполнение следующих трёх этапов.

1. Сначала за каждой буквой входного слова помещаем символ | (палочка).
2. Затем перемещаем все палочки вправо. В результате они окажутся правее исходного слова, причём их количество будет равно количеству букв.
3. На последнем этапе удаляем комбинации, состоящие из буквы и двух палочек (то есть комбинации $a||$ и $b||$), пока это возможно. С помощью этого приёма мы избавляемся от ненужных нам букв правой половины слова (ведь каждый раз удаляются две палочки и только одна самая правая буква).

Все указанные действия описываются в виде следующего НАМ:

выписано столько палочек, какова величина числа; например: $2 \rightarrow ||$, $5 \rightarrow |||||$, $0 \rightarrow \langle \text{пустое слово} \rangle$.

2.1 $A = \{f, h, p\}$. В слове P заменить все пары ph на f .

2.2 $A = \{f, h, p\}$. В слове P заменить на f только первую пару ph , если такая есть.

2.3 $A = \{a, b, c\}$. Приписать слово bac слева к слову P .

2.4 $A = \{a, b, c\}$. Заменить слово P на пустое слово, т.е. удалить из P все символы.

2.5 $A = \{a, b, c\}$. Заменить любое входное слово на слово a .

2.6 Построить НАМ, не меняющий входное слово (при любом алфавите A).

2.7 $A = \{ | \}$. Считая слово P записью числа в единичной системе счисления, получить остаток от деления этого числа на 2, т.е. получить слово из одной палочки, если число нечётно, или пустое слово, если число чётно.

2.8 $A = \{ | \}$. Считая слово P записью положительного числа в единичной системе счисления, уменьшить это число на 1.

2.9 $A = \{ | \}$. Считая слово P записью числа в единичной системе счисления, увеличить это число на 2.

2.10 $A = \{0, 1, 2\}$. Считая слово P записью числа в троичной системе счисления, получить остаток от деления этого числа на 2, т.е. получить слово 1, если число нечётно, или слово 0, если число чётно. (Замечание: в чётном троичном числе должно быть чётное количество цифр 1.)

2.11 $A = \{a, b, c\}$. Определить, входит ли символ a в слово P . Ответ (выходное слово): слово a , если входит, или пустое слово, если не входит.

2.12 $A = \{a, b\}$. Если в слово P входит больше символов a , чем символов b , то в качестве ответа выдать слово из одного символа a , если в P равное количество a и b , то в качестве ответа выдать пустое слово, а иначе выдать ответ b .

2.13 $A = \{0, 1, 2, 3\}$. Преобразовать слово P так, чтобы сначала шли все чётные цифры (0 и 2), а затем – все нечётные.

- 2.14 $A=\{a,b,c\}$. Преобразовать слово P так, чтобы сначала шли все символы a , затем – все символы b и в конце – все символы c .
- 2.15 $A=\{a,b,c\}$. Определить, из скольких различных символов составлено слово P ; ответ получить в единичной системе счисления (например: $acaac \rightarrow ||$).
- 2.16 $A=\{a,b,c\}$. В непустом слове P удвоить первый символ, т.е. приписать этот символ слева к P .
- 2.17 $A=\{a,b,c\}$. За первым символом непустого слова P вставить символ c .
- 2.18 $A=\{a,b,c\}$. Из слова P удалить второй символ, если такой есть.
- 2.19 $A=\{a,b,c\}$. Если в слове P не менее двух символов, то переставить два первых символа.
- 2.20 $A=\{0,1,2\}$. Считая непустое слово P записью троичного числа, удалить из этой записи все незначащие нули.
- 2.21 $A=\{a,b,c\}$. Приписать слово abc справа к слову P .
- 2.22 $A=\{a,b,c\}$. Удалить из непустого слова P его последний символ.
- 2.23 $A=\{0,1\}$. Считая непустое слово P записью числа в двоичной системе, получить двоичное число, равное учетверённому числу P (например: $101 \rightarrow 10100$).
- 2.24 $A=\{0,1\}$. Считая непустое слово P записью числа в двоичной системе, получить двоичное число, равное неполному частному от деления числа P на 2 (например: $1011 \rightarrow 101$).
- 2.25 $A=\{a,b\}$. В слове P все символы a заменить на b , а все (прежние) символы b – на a .
- 2.26 $A=\{a,b,c\}$. Удвоить каждый символ в слове P (например: $bacb \rightarrow bbaaccbb$).
- 2.27 $A=\{a,b\}$. Приписать слева к слову P столько палочек, сколько всего символов входит в P (например: $babb \rightarrow ||| |babb$).
- 2.28 $A=\{a,b\}$. Пусть слово P имеет чётную длину (0, 2, 4, ...). Удалить левую половину этого слова. (*Рекомендация*: использовать решение предыдущей задачи.)
- 2.29 $A=\{a,b\}$. Пусть длина слова P кратна 3. Удалить правую треть этого слова.

- 2.30 $A=\{a,b\}$. Приписать справа к слову P столько палочек, со скольких подряд идущих символов a начинается это слово (например: $aababa \rightarrow aababa| |$).
- 2.31 $A=\{a,b,c\}$. Удалить из слова P второе вхождение символа a , если такое есть.
- 2.32 $A=\{a,b,c\}$. Удалить из слова P третье вхождение символа a , если такое есть.
- 2.33 $A=\{a,b,c\}$. Оставить в слове P только первое вхождение символа a , если такое есть.
- 2.34 $A=\{a,b,c\}$. В непустом слове P оставить только последний символ.
- 2.35 $A=\{a,b,c\}$. Из всех вхождений символа a в слово P оставить только последнее вхождение, если такое есть.
- 2.36 $A=\{a,b,c\}$. Если слово P начинается с символа a , то заменить P на пустое слово, а иначе P не менять.
- 2.37 $A=\{a,b\}$. Если слово P содержит одновременно символы a и b , то заменить P на пустое слово.
- 2.38 $A=\{a,b,c\}$. Если буквы в непустом слове P не упорядочены по алфавиту, то заменить P на пустое слово, а иначе P не менять.
- 2.39 $A=\{a,b,c\}$. Если P отлично от слова $abaca$, то заменить его на пустое слово.
- 2.40 $A=\{0,1\}$. Считая непустое слово P записью двоичного числа, определить, является ли это число степенью 2 (1, 2, 4, ...). Ответ: слово 1, если является, или слово 0 иначе.
- 2.41 $A=\{0,1,2,3\}$. Считая непустое слово P записью четверичного числа, проверить, чётно оно или нет. Ответ: слово 0, если чётно, и слово 1 иначе.
- 2.42 $A=\{0,1,2,3\}$. Считая непустое слово P записью четверичного числа, получить остаток от деления этого числа на 4.
- 2.43 $A=\{0,1\}$. Считая непустое слово P записью двоичного числа, получить это же число, но в четверичной системе. (Замечание: учесть, что в двоичном числе может быть нечётное количество цифр.)
- 2.44 $A=\{0,1,2\}$. Считая непустое слово P записью троичного числа, увеличить это число на 1.

2.45 $A=\{0,1,2\}$. Считая непустое слово P записью положительного троичного числа, уменьшить это число на 1.

2.46 $A=\{ | \}$. Считая слово P записью числа в единичной системе счисления, получить запись этого числа в троичной системе. (Рекомендация: следует в цикле удалять из «единичного» числа по палочке и каждый раз прибавлять 1 к троичному числу, которое вначале положить равным 0.)

2.47 $A=\{0,1,2\}$. Считая непустое слово P записью числа в троичной системе, получить запись этого числа в единичной системе.

2.48 $A=\{a,b,c\}$. Определить, входит ли первый символ непустого слова P ещё раз в это слово. Ответ: слово a , если входит, или пустое слово иначе.

2.49 $A=\{a,b\}$. Перенести первый символ непустого слова P в конец слова.

2.50 $A=\{a,b\}$. Перенести последний символ непустого слова P в начало слова.

2.51 $A=\{a,b\}$. В непустом слове P переставить первый и последний символы.

2.52 $A=\{a,b\}$. Если в непустом слове P совпадают первый и последний символы, то удалить оба этих символа, а иначе слово не менять.

2.53 $A=\{a,b\}$. Определить, является ли слово P палиндромом (перевёртышем, симметричным словом). Ответ: слово a , если является, или пустое слово иначе.

2.54 $A=\{a,b\}$. Пусть слово P имеет нечётную длину. Удалить из него средний символ.

2.55 Пусть слово P имеет следующий вид:

$$\underbrace{||\dots|}_{n} \otimes \underbrace{||\dots|}_{m}$$

где \otimes – один из знаков $+$, $-$, \times , $/$, \div , \uparrow или \downarrow , слева от которого указано n палочек, а справа – m палочек. Реализовать соответствующую операцию в единичной системе счисления (в качестве ответа выдать слово, указанное справа от стрелки):

а) сложение: $\underbrace{||\dots|}_{n} + \underbrace{||\dots|}_{m} \rightarrow \underbrace{||\dots|}_{n+m} \quad (n \geq 0, m \geq 0)$

б) вычитание: $\underbrace{||\dots|}_{n} - \underbrace{||\dots|}_{m} \rightarrow \underbrace{||\dots|}_{n-m} \quad (n \geq m \geq 0)$

в) умножение: $\underbrace{||\dots||}_n \times \underbrace{||\dots||}_m \rightarrow \underbrace{||\dots||}_{n \times m} \quad (n \geq 0, m \geq 0)$

г) деление нацело: $\underbrace{||\dots||}_n / \underbrace{||\dots||}_m \rightarrow \underbrace{||\dots||}_k \quad (n \geq 0, m > 0, k = n \operatorname{div} m)$

д) взятие остатка: $\underbrace{||\dots||}_n \div \underbrace{||\dots||}_m \rightarrow \underbrace{||\dots||}_k \quad (n \geq 0, m > 0, k = n \operatorname{mod} m)$

е) максимум: $\underbrace{||\dots||}_n \uparrow \underbrace{||\dots||}_m \rightarrow \underbrace{||\dots||}_k \quad (n \geq 0, m \geq 0, k = \max(n, m))$

ж) минимум: $\underbrace{||\dots||}_n \downarrow \underbrace{||\dots||}_m \rightarrow \underbrace{||\dots||}_k \quad (n \geq 0, m \geq 0, k = \min(n, m))$

2.56 $A = \{ | \}$. Считая слово P записью числа в единичной системе, определить, является ли это число степенью 3 (1, 3, 9, 27, ...). Ответ: пустое слово, если является, или слово из одной палочки иначе.

2.57 $A = \{ | \}$. Считая слово P записью числа n в единичной системе, получить в этой же системе число 2^n .

2.58 $A = \{ | \}$. Пусть слово P является записью числа 2^n ($n = 0, 1, 2, \dots$) в единичной системе. Получить в этой же системе число n .

2.59 Пусть P имеет вид $Q+R$, где Q и R – непустые слова из символов 0, 1 и 2. Трактую Q и R как записи троичных чисел (возможно, с незначащими нулями), выдать в качестве ответа запись суммы этих чисел в той же троичной системе.

2.60 Пусть P имеет вид $Q-R$, где Q и R – непустые слова из символов 0, 1 и 2. Трактую Q и R как записи неотрицательных троичных чисел (возможно, с незначащими нулями) и считая, что $Q \geq R$, выдать в качестве ответа запись разности этих чисел в той же троичной системе.

2.61 Пусть P имеет вид $Q=R$, где Q и R – любые слова из символов a и b . Выдать ответ a , если слова Q и R одинаковы, и пустое слово иначе.

2.62 Пусть P имеет вид $Q=R$, где Q и R – непустые слова из символов 0 и 1. Трактую Q и R как записи двоичных чисел (возможно, с незначащими нулями), выдать в качестве ответа слово 1, если эти числа равны, и слово 0 иначе.

2.63 Пусть P имеет вид $Q>R$, где Q и R – непустые слова из символов 0 и 1. Трактую Q и R как записи двоичных чисел (возможно, с незначащими нулями), выдать в качестве ответа слово 1, если число Q больше числа R , и слово 0 иначе.

2.64 $A = \{ (,) \}$. Определить, сбалансировано ли слово P по круглым скобкам. Ответ: D (да) или H (нет)

2.65 $A=\{a,b\}$. Перевернуть слово P (например: $abb \rightarrow bba$).

2.66 $A=\{ | \}$. С помощью пары прямоугольных скобок отметить середину последовательности из палочек. Примеры: $||| \rightarrow |[]|$, $|||| \rightarrow |[]| |$, $| \rightarrow []$, $\langle \text{пусто} \rangle \rightarrow []$.

2.67 Сформулировать условие задачи, которую решает НАМ над алфавитом $\{ |, \# \}$ в применении к словам вида $x \# y$, где x и y – натуральные числа в единичной системе счисления. НАМ задается следующим набором формул:

$$\left\{ \begin{array}{l} |a \rightarrow a| \\ |\#| \rightarrow a\# \\ |\# \rightarrow \#b \\ b \rightarrow | \\ a \rightarrow c \\ c \rightarrow | \\ \# \mapsto \end{array} \right.$$

2.68 Сформулировать условие задачи, которую решает НАМ над алфавитом $\{ |, \# \}$ в применении к словам вида $x \# y$, где x и y – натуральные числа в единичной системе счисления. НАМ задается следующим набором формул:

$$\left\{ \begin{array}{l} b| \rightarrow |b \\ a| \rightarrow |ba \\ a \rightarrow \\ |\# \rightarrow \#a \\ \#| \rightarrow \# \\ \# \rightarrow \\ b \rightarrow | \end{array} \right.$$

2.69 Найти максимум среди последовательности чисел в единичной системе счисления. Числа в исходном слове разделены решёткой, ответ – число в единичной системе. Примеры: $|\#\#||\#\#| \rightarrow ||$, $||||\#\#||\#\#|| \rightarrow ||||$, $\langle \text{пусто} \rangle \rightarrow \langle \text{пусто} \rangle$.

2.70 Число n , записанное в единичной системе счисления, перевести в число $\lfloor \log_2 n \rfloor$. Примеры: $|||| \rightarrow ||$, $||| \rightarrow ||$, $||||| \rightarrow ||$, $| \rightarrow \langle \text{пусто} \rangle$.

2.71 Построить НАМ, применимый к тем и только тем словам в алфавите $\{0,1\}$, которые одновременно удовлетворяют следующим условиям: а) в слове есть хотя бы две цифры 0; б) общее число нулей меньше, чем число единиц. НАМ должен содержать не более 5 формул.

2.72 Построить НАМ, применимый к тем и только тем словам из алфавита $\{0,1\}$, которые одновременно удовлетворяют следующим условиям: а) слово содержит не менее трех цифр; б) цифры 0 и 1 чередуются. Например, НАМ должен быть применим к словам 101, 01010, и неприменим к словам 10, 10111.

2.73 Построить НАМ, применимый к тем и только тем словам из алфавита $\{a,b\}$, которые одновременно удовлетворяют следующим условиям: а) слово содержит не менее трех букв; б) в слове присутствуют либо только буквы a , либо только буквы b . Например, НАМ должен быть применим к словам aaa , $bbbb$, и неприменим к словам aa , $aaab$.

2.74 Построить НАМ, применимый к тем и только тем словам из алфавита $\{a,b\}$, которые удовлетворяют хотя бы одному из следующих условий: а) число букв a в слове чётно; б) число букв b в слове кратно 3. НАМ должен содержать не более 5 формул.

3. Задачи теоретического характера

В разделе рассматриваются несложные задачи по теории алгоритмов. При этом приводятся необходимые сведения из этой теории. В качестве алгоритмов в основном используются нормальные алгоритмы Маркова (НАМ).

В разделе применяются следующие обозначения:

1. Последовательность из n подряд идущих символов a будем обозначать как a^n ; например: a^0 – это пустое слово, a^1 – это a , a^4 – это $aaaa$ и т.д.
2. Фраза «слово в алфавите A » означает, что в слово входят лишь символы из A .
3. Если алгоритм H применим к слову P , тогда результат применения H к P (выходное слово) будем обозначать как $H(P)$.

3.1 Применимость алгоритма

Напомним, что алгоритм называется *применимым* к слову, если, начав работать над этим словом как входным, он остановится через конечное число шагов. Если же алгоритм зацикливается, то он *неприменим* к этому слову.

Отметим, что когда просят определить, применим алгоритм к слову или нет, то требуется лишь указать, остановится ли алгоритм, и не требуется указывать, каков результат применения алгоритма к слову.

Область применимости алгоритма относительно некоторого алфавита – это множество всех таких слов в этом алфавите, к которым применим алгоритм.

Пример 1

Определить область применимости следующего НАМ относительно алфавита $\{a,b\}$:

$$\begin{cases} b \rightarrow b & (1) \\ a \mapsto b & (2) \end{cases}$$

(Напомним, что номера не входят в формулы, а используются лишь для ссылок на них.)

Решение

Формула (1) применима к любому входному слову, в которое входит хотя бы один символ b , причём она не меняет это слово. Поэтому на таких словах данный НАМ зацикливается. Если же во входном слове нет символов b , но есть хотя бы один символ a , тогда формула (1) не будет работать, а сработает заключительная формула (2), которая остановит алгоритм. Следовательно, НАМ останавливается на словах, состоящих только из символов a . Что же касается пустого входного слова, то в этом случае обе формулы неприменимы, поэтому НАМ сразу остановится.

Итак, область применимости указанного НАМ – все слова вида a^n ($n \geq 0$).

Пример 2

Построить НАМ, который применим ко всем словам в алфавите $\{a,b,c\}$, кроме двух слов – abc и $baac$.

Решение

Из условия задачи следует, что НАМ должен заикливаться на двух указанных словах. Однако это не значит, что задачу решает следующий НАМ:

$$\begin{cases} abc \rightarrow abc \\ baac \rightarrow baac \end{cases}$$

Дело в том, что данный алгоритм заикливается не только на этих двух словах, но и на любых словах, в которые они входят как подслова, например, на слове *ccabcbb*.

Поэтому надо как-то отличать случай, когда каждое из указанных слов целиком составляет входное слово, от случая всех остальных слов. Обычно в такой ситуации поступают следующим образом: начало и конец входного слова P помечают какими-то спецзнаками (например, $*P\#$), а затем используют формулы, в левой части которых указывают нужные слова и эти спецзнаки ($*abc\#$ и $*baac\#$). Такие формулы будут применимы только к нужным входным словам.

В нашем примере эти формулы должны заикливать алгоритм, а для останова его на других словах можно использовать, например, формулу $* \mapsto$. Итого, получаем:

$$\begin{cases} \#a \rightarrow a\# \\ \#b \rightarrow b\# \\ \#c \rightarrow c\# \\ *abc\# \rightarrow *abc\# \\ *baac\# \rightarrow *baac\# \\ * \mapsto \\ \rightarrow *\# \end{cases}$$

3.2 Самоприменимость алгоритма

В первом приближении самоприменимым называют алгоритм (скажем, НАМ), который применим к самому себе. Однако это некорректное определение, поскольку на вход НАМ можно подавать только слова (линейные последовательности символов), а НАМ таковым не является. Поэтому, чтобы сделать это определение корректным, надо как-то «вытянуть» НАМ в линию. Для этого вводится

понятие записи алгоритма: *записью* НАМ называется слово, состоящее из последовательно записанных через точку с запятой формул подстановки этого алгоритма (точки с запятой отделяют друг от друга соседние формулы). При этом предполагается, что точки с запятой не входят в формулы; если это не так, то вместо точки с запятой можно использовать любой другой символ, не входящий в формулы.

Пусть, к примеру, имеются следующие алгоритмы H_1 и H_2 :

$$H_1: \begin{cases} \#a \rightarrow \# & (1) \\ \# \mapsto & (2) \\ \rightarrow \# & (3) \end{cases} \quad H_2: \begin{cases} a \rightarrow b & (4) \\ b \rightarrow bb & (5) \end{cases}$$

Тогда записью алгоритма H_1 является слово

$$\#a \rightarrow \# ; \# \mapsto ; \rightarrow \#$$

а записью алгоритма H_2 – слово

$$a \rightarrow b ; b \rightarrow bb$$

Легко понять, что алгоритм однозначно определяет свою запись и наоборот. Учитывая это, а также то, что запись алгоритма является словом, которое уже можно подавать на вход алгоритму, в указанном выше определении нужно заменить фразу «применим к самому себе» на фразу «применим к своей записи». В результате получаем следующее корректное определение самоприменимости: алгоритм называется *самоприменимым*, если он применим к своей записи, и *несамоприменимым* в противном случае.

Например, алгоритм H_1 самоприменим, т.к. начав работать над своей записью как входным словом, он остановится:

$$\# \underline{a} \rightarrow \# ; \# \mapsto ; \rightarrow \# \xrightarrow{1} \# \rightarrow \# ; \# \mapsto ; \rightarrow \# \xrightarrow{2} \mapsto ; \# \mapsto ; \rightarrow \#$$

Алгоритм же H_2 несамоприменим, т.к. он заикликивается на своей записи:

$$\underline{a} \rightarrow b ; b \rightarrow bb \xrightarrow{4} \underline{b} \rightarrow b ; b \rightarrow bb \xrightarrow{5} \underline{bb} \rightarrow b ; b \rightarrow bb \xrightarrow{5} \dots$$

Отметим, что если применимость зависит как от алгоритма, так и от слова, то самоприменимость зависит только от алгоритма, от того, применим ли этот алгоритм к конкретному слову – к своей собственной записи, которая однозначно определяется данным алгоритмом.

Пример 3

Построить НАМ, который несамоприменим, но применим к любому слову в алфавите $\{a,b\}$.

Решение

Поскольку искомый НАМ применим ко всем словам, составленным из символов a и b , то зацикливаться он может лишь на словах, содержащих какой-то дополнительный символ, скажем $*$. Следовательно, чтобы НАМ оказался несамоприменимым (зацикливался на своей записи), символ $*$ должен входить в запись алгоритма (в одну из его формул) и на этом символе НАМ должен циклиться. Можно придумать много таких алгоритмов, но простейшим из них является следующий:

$$\{ * \rightarrow *$$

Пример 4

Известно, что проблема самоприменимости алгоритмически неразрешима, т.е. не существует единого способа (алгоритма), который позволял бы определять для любого алгоритма, самоприменим он или нет. Однако в частных случаях (для некоторых классов алгоритмов) эта проблема может оказаться и разрешимой. Например, проблема самоприменимости для НАМ, содержащих только одну формулу подстановки, разрешима. Требуется доказать это утверждение.

Доказательство

Пусть НАМ имеет вид $\{ \alpha \mapsto \beta$ или $\{ \alpha \rightarrow \beta$. Тогда можно использовать следующий метод проверки самоприменимости: если единственная формула алгоритма заключительная или если это обычная формула, левая часть (α) которой не входит в правую часть (β), то такой НАМ самоприменим, иначе же он несамоприменим.

Действительно, какой бы ни была единственная формула – заключительной или обычной, на первом шаге применения НАМ к его записи (к слову $\alpha \mapsto \beta$ или $\alpha \rightarrow \beta$) часть α в этой записи будет заменена на β , в результате чего получится слово $\beta \mapsto \beta$ или $\beta \rightarrow \beta$. Если формула заключительная, то на этом НАМ и остановится, а это значит, что он самоприменим. Если же формула обычная и α не входит в β , то формула неприменима к получившемуся слову и работа НАМ

прекращается, поэтому и в данном случае алгоритм самоприменим. Но если формула обычная и α входит в β , то в получившемся слове $\beta \rightarrow \beta$ будет присутствовать α , поэтому наша формула снова применяется, причем в новом слове по-прежнему окажется α . Получаем бесконечный процесс подстановок, а это и значит, что НАМ несамоприменим.

3.3 Эквивалентность алгоритмов

Эквивалентными называются алгоритмы, которые предлагают разные способы решения одной и той же задачи. Это значит, что на одинаковых входных словах они выдают одинаковые результаты (выходные слова). При этом, правда, надо учитывать и случай заикливания: если один алгоритм заикливается на каких-то входных словах, т.е. не даёт решение задачи, то и эквивалентный ему алгоритм также не должен давать решения на этих исходных данных.

Точное определение эквивалентности алгоритмов следующее: алгоритмы H_1 и H_2 *эквивалентны относительно алфавита A* , если области применимости H_1 и H_2 относительно алфавита A совпадают и для любого слова P из этой области выполняется равенство $H_1(P) = H_2(P)$.

Отметим, что в этом определении важен тот алфавит, относительно которого устанавливается эквивалентность. Дело в том, что одни и те же алгоритмы могут быть эквивалентными относительно одного алфавита и не эквивалентными относительно другого алфавита. Например, алгоритмы

$$H_1 : \{ a \mapsto a \} \qquad H_2 : \begin{cases} a \mapsto a \\ b \rightarrow b \end{cases}$$

эквивалентны относительно алфавита $\{a\}$ и не эквивалентны относительно алфавита $\{a,b\}$: слова в алфавите $\{a\}$ они не меняют, но если входное слово состоит только из символов b , то H_1 остановится, а H_2 заиклится.

Пример 5

Определить, эквивалентны ли следующие пары НАМ относительно алфавита $\{a,b\}$:

$$\begin{array}{ll}
1) H_1 : \{ a \mapsto b & H_2 : \left\{ \begin{array}{l} *b \rightarrow b* \\ *a \mapsto b \\ \rightarrow * \end{array} \right. \\
2) H_3 : \left\{ \begin{array}{l} a \rightarrow b \\ b \rightarrow a \end{array} \right. & H_4 : \left\{ \begin{array}{l} a \rightarrow aa \\ b \rightarrow bb \end{array} \right. \\
3) H_5 : \{ aa \rightarrow a & H_6 : \left\{ \begin{array}{l} *aa \rightarrow a* \\ *b \rightarrow b* \\ * \mapsto \\ \rightarrow * \end{array} \right.
\end{array}$$

Решение

При проверке алгоритмов на эквивалентность надо прежде всего установить, совпадают ли области их применимости.

В случае алгоритмов H_1 и H_2 , которые первое вхождение символа a заменяют на b , это условие не выполняется: H_1 применим ко всем словам из символов a и b , а H_2 заикливаются на словах, не содержащих a . Значит, алгоритмы H_1 и H_2 не эквивалентны.

Если же области применимости совпадают, тогда надо показать, что на одних и тех же словах из данной области эти алгоритмы выдают одинаковые результаты.

У пары алгоритмов H_3 и H_4 одна и та же область применимости – она состоит только из одного пустого слова. На непустых же словах эти алгоритмы заикливаются, причём заикливаются по-разному: H_3 постоянно меняет a на b и b на a , тогда как H_4 всё время добавляет новый символ a или b . Однако такое различное поведение при заиклипании не играет никакой роли при определении эквивалентности алгоритмов. Важно лишь, чтобы в случае останова алгоритмы выдавали одинаковые выходные слова. А для пары H_3 и H_4 это условие выполняется: на единственном слове (пустом), к которому они применимы, они выдают один и тот же ответ – пустое слово. Значит, алгоритмы H_3 и H_4 эквивалентны.

Теперь рассмотрим пару алгоритмов H_5 и H_6 . У них одна и та же область применимости – это множество всех слов в алфавите $\{a,b\}$. Однако второе условие эквивалентности (одинаковые результаты при одинаковых исходных данных) не выполняется. Чтобы доказать это,

достаточно привести лишь одно слово, на котором алгоритмы выдают разные ответы. Таким словом может быть, например, слово $aaaa$:

$$H_5: \underline{aaaa} \rightarrow \underline{aaa} \rightarrow \underline{aa} \rightarrow a$$

$$H_6: aaaa \rightarrow \underline{*aaaa} \rightarrow a\underline{*aa} \rightarrow aa\underline{*} \mapsto aa$$

Итак, алгоритмы H_5 и H_6 не эквивалентны.

3.4 Композиция алгоритмов

Как известно, к результату одной функции можно применить другую функцию, например: $\sin(\operatorname{ctg} x)$. Точно так же выходное слово одного алгоритма H_1 можно подать на вход другому алгоритму H_2 . Такое последовательное выполнение сначала алгоритма H_1 , а затем алгоритма H_2 называется композицией этих алгоритмов. При этом, правда, надо учитывать, что любой из этих алгоритмов может заикливаться, тогда должна заикливаться и их композиция.

Эти соображения приводят к следующему определению:

Композицией алгоритмов H_1 и H_2 относительно алфавита A называется такой алгоритм H (обозначается как $H_2(H_1)$ или $H_1 \circ H_2$), что для любого слова P в алфавите A выполняются следующие условия:

- 1) если H_1 неприменим к P , то и H неприменим к P ;
- 2) если H_1 применим к P , но H_2 неприменим к слову $H_1(P)$, то и H неприменим к P ;
- 3) если H_1 применим к P и H_2 применим к слову $H_1(P)$, то H применим к P , причём выполняется равенство $H(P) = H_2(H_1(P))$.

Доказана следующая теорема: для любых нормальных алгоритмов H_1 и H_2 существует нормальный алгоритм H , который является (относительно соответствующего алфавита) композицией H_1 и H_2 . (Аналогичная теорема верна и для машин Тьюринга.)

Отметим, что на самом деле существует способ построения списка формул для композиции H по спискам формул из H_1 и H_2 (наличие этого способа, кстати, и доказывает указанную выше теорему). Формальное описание этого способа изложено в [3]. Мы же проиллюстрируем применение этого способа на конкретном примере, отражая все необходимые приемы. По аналогии с этим примером можно успешно строить композицию любых двух алгоритмов.

Пример 6 (построение композиции алгоритмов)

Построить нормальный алгоритм H , являющийся композицией нормальных алгоритмов H_1 и H_2 относительно алфавита $\{a,b\}$, т.е. $H = H_1 \circ H_2$:

$$H_1 : \{ abb \mapsto a \qquad H_2 : \{ b \mapsto$$

Решение

При первом, поверхностном взгляде может показаться, что композицией H является последовательное выписывание формул подстановки одного алгоритма вслед за формулами подстановки другого алгоритма. Но даже наш несложный пример опровергает такой подход. Легко видеть, что последовательное выписывание формул исходных алгоритмов (в любом порядке: H_1 вслед за H_2 , H_2 вслед за H_1) не является представлением их композиции (самостоятельно проверьте это, например, на входном слове $aabaabbbba$).

Для построения искомой композиции H применим к нашим алгоритмам H_1 и H_2 универсальный способ, основанный на методе из [3].

Сначала преобразуем H_1 и H_2 следующим образом. Заметим, что если в конец записи любого алгоритма приписать заключительную формулу \mapsto (с пустыми левой и правой частями), то получится эквивалентный алгоритм, обладающий свойством: на любом входном слове, к которому он применим, этот алгоритм завершается выполнением заключительной формулы подстановки. Добавим в конец наших алгоритмов H_1 и H_2 заключительную формулу с пустыми левой и правой частями:

$$H_1 : \begin{cases} abb \mapsto a \\ \mapsto \end{cases} \qquad H_2 : \begin{cases} b \mapsto \\ \mapsto \end{cases}$$

Теперь каждый из этих алгоритмов на любом входном слове (в том числе и пустом) останавливается только по заключительной формуле.

Композиция преобразованных алгоритмов H_1 и H_2 осуществляется по нижеприведенному плану, состоящему из пяти пунктов. Каждый пункт мы прокомментируем по отношению к H_1 и H_2 . (Затем, после всех замечаний и комментариев, будет дан окончательный список формул для получившегося алгоритма-композиции H).

1. В композиции первыми будем выполнять формулы подстановки из H_1 . Далее (вместо завершения) перейдем к выполнению формул из H_2 . Для «стыковки» последовательной работы алгоритмов заменим в H_1 все заключительные формулы на обычные с приписыванием спецзнака l в начало правой части каждой такой формулы, т.е. формулы вида $\alpha \mapsto \beta$ заменим на формулы вида $\alpha \rightarrow l\beta$. В нашем примере формулы из H_1 преобразуются в такие:

$$\left. \begin{array}{l} abb \rightarrow la \\ \rightarrow l \end{array} \right\} \text{изменённый } H_1$$

Отметим, что появление единицы в обрабатываемом слове сигнализирует об окончании H_1 и необходимости перехода к выполнению H_2 .

2. Прежде чем начать работу H_2 , заблокируем выполнение всех формул из H_1 (это важно, так как оба НАМ определены в одном алфавите). Для этого перегоним единицу влево, после чего выполним процесс перекодирования символов текущего слова в алфавит двойников – все строчные буквы заменим на прописные. (В нашем примере алфавит двойников – это множество $\{A, B\}$). Эти действия обеспечиваются выполнением следующих формул:

$$\left. \begin{array}{l} al \rightarrow la \\ bl \rightarrow lb \\ la \rightarrow lA \\ lb \rightarrow lB \\ Aa \rightarrow AA \\ Ab \rightarrow AB \\ Ba \rightarrow BA \\ Bb \rightarrow BB \end{array} \right\} \begin{array}{l} \text{перемещение единицы влево} \\ \text{старт процесса перекодирования} \\ \text{перекодирование в алфавит двойников} \end{array}$$

3. Преобразуем теперь алгоритм H_2 следующим образом. Заменим в его формулах подстановки все символы на их двойников, а заключительные формулы вида $\alpha \mapsto \beta$ заменим на обычные формулы вида $\alpha \rightarrow 2\beta$. Таким образом, получим формулы:

$$\left. \begin{array}{l} B \rightarrow 2 \\ \rightarrow 2 \end{array} \right\} \text{алгоритм } H_2 \text{ в алфавите двойников с} \\ \text{изменёнными заключительными} \\ \text{формулами}$$

Теперь в преобразованном алгоритме все формулы подстановки вида $\rightarrow \beta$ (с пустой левой частью) заменим на формулы вида $l \rightarrow l\beta$. Получим:

$$\left. \begin{array}{l} B \rightarrow 2 \\ l \rightarrow l2 \end{array} \right\} \text{окончательно изменённый } H_2$$

Отметим, что появление двойки в обрабатываемом слове сигнализирует об окончании H_2 и необходимости возврата к исходному алфавиту.

4. Перегоним двойку влево до единицы, после чего выполним процесс перекодирования символов-двойников обратно в исходные (в нашем примере все прописные буквы A и B заменяются на строчные). Для этого понадобятся формулы:

$$\left. \begin{array}{l} A2 \rightarrow 2A \\ B2 \rightarrow 2B \\ 2A \rightarrow 2a \\ 2B \rightarrow 2b \\ aA \rightarrow aa \\ aB \rightarrow ab \\ bA \rightarrow ba \\ bB \rightarrow bb \end{array} \right\} \begin{array}{l} \text{перемещение двойки влево} \\ \\ \text{старт обратного перекодирования} \\ \\ \text{перекодирование из двойников} \\ \text{в исходные символы} \end{array}$$

5. После обратного перекодирования удалим пару спецзнаков $l2$ и на этом завершим работу алгоритма-композиции.

$$l2 \mapsto \quad] \text{останов}$$

Пусть H – композиция алгоритмов H_1 и H_2 , построенная по нашему плану. Проиллюстрируем пошаговую работу алгоритма H в применении к входному слову $P = aabaabbbba$ и убедимся, что $H(P) = H_2(H_1(P))$:

$$\left. \begin{array}{l} aabaabbbba \\ aaba\underline{l}abba \\ aab\underline{1}aabba \\ a\underline{a}1baabba \\ \underline{a}1abaabba \\ \underline{1}aabaabba \\ 1\underline{A}abaabba \\ 1\underline{A}A\underline{a}abba \end{array} \right\} \begin{array}{l} \text{применение } H_1 \text{ к входному слову} \\ \text{появление единицы сигнализирует об окончании } H_1 \\ \\ \text{перемещение единицы влево} \\ \\ \text{единица слева – сигнал к перекодированию в алфавит} \\ \text{двойников} \\ \text{перекодирование (замена строчных букв на прописные)} \end{array}$$

...

...	
$1A\underline{A}BAABBA$	перекодирование (замена строчных букв на прописные)
$1AA\underline{2}AABBA$	
$1A\underline{2}AAABBA$	выполнение H_2 в алфавите двойников
$1\underline{2}AAAABBA$	
$12\underline{a}AAABBA$	появление двойки – сигнал окончания H_2
$12a\underline{a}AABBA$	
...	перемещение двойки влево до единицы
$\underline{1}2aaaabba$	
$aaaabba$	пара 12 слева – сигнал к возврату в исходный алфавит
	обратное перекодирование в исходный алфавит
	возврат в исходный алфавит завершен
	последний этап: удаление спецзнаков и останов

Приведем теперь полностью готовую композицию H (с размещенными в нужном порядке группами формул):

$H:$	$a1 \rightarrow 1a$	(1)		
	$b1 \rightarrow 1b$	(2)		
	$1a \rightarrow 1A$	(3)		
	$1b \rightarrow 1B$	(4)	2. Перемещение единицы влево и перекодирование символов в их двойники	
	$Aa \rightarrow AA$	(5)		
	$Ab \rightarrow AB$	(6)		
	$Ba \rightarrow BA$	(7)		
	$Bb \rightarrow BB$	(8)		

	$A2 \rightarrow 2A$	(9)		
	$B2 \rightarrow 2B$	(10)		
	$2A \rightarrow 2a$	(11)	4. Перемещение двойки влево до единицы и обратное перекодирование в исходный алфавит	
	$2B \rightarrow 2b$	(12)		
	$aA \rightarrow aa$	(13)		
	$aB \rightarrow ab$	(14)		
	$bA \rightarrow ba$	(15)		
	$bB \rightarrow bb$	(16)		

	$12 \mapsto$	(17)	5. Останов	

	$B \rightarrow 2$	(18)	3. Преобразованный алгоритм H_2 (в алфавите двойников)	
$1 \rightarrow 12$	(19)			

$abb \rightarrow 1a$	(20)	1. Преобразованный алгоритм H_1		
$\rightarrow 1$	(21)			

Приведем последовательность применения формул построенного алгоритма H к нашему слову $P = aabaabbbba$:

$$\begin{aligned}
 & aaba\underline{abbb}ba \xrightarrow{20} aaba\underline{1}abba \xrightarrow{1} aab\underline{1}aabba \xrightarrow{2} aa\underline{1}baabba \xrightarrow{1} \\
 & \rightarrow \underline{a}1abaabba \xrightarrow{1} \underline{1}aabaabba \xrightarrow{3} 1\underline{A}abaabba \xrightarrow{5} 1A\underline{A}baabba \xrightarrow{6} \\
 & \rightarrow 1A\underline{A}Baabba \xrightarrow{7} 1AAB\underline{A}abba \xrightarrow{5} 1AABA\underline{A}bba \xrightarrow{6} 1AABAAB\underline{b}a \xrightarrow{8} \\
 & \rightarrow 1AABAAB\underline{B}a \xrightarrow{7} 1AABAAB\underline{B}A \xrightarrow{18} 1A\underline{A}2AABBA \xrightarrow{9} 1A\underline{2}AAABBA \xrightarrow{9} \\
 & \rightarrow 1\underline{2}AAAABBA \xrightarrow{11} 12\underline{a}AAABBA \xrightarrow{13} 12aa\underline{A}ABBA \xrightarrow{13} 12aaa\underline{A}BBA \xrightarrow{13} \\
 & \rightarrow 12aaaa\underline{B}BA \xrightarrow{14} 12aaaa\underline{ab}BA \xrightarrow{16} 12aaaa\underline{abb}A \xrightarrow{15} \underline{1}2aaaa\underline{ab}ba \xrightarrow{17} \\
 & \mapsto aaaaabba
 \end{aligned}$$

Попробуйте самостоятельно проиллюстрировать работу H на пустом слове, на слове без вхождения abb , и на слове, не содержащем b .

Как видим, построенный в соответствии с формальным способом алгоритм $H=H_2(H_1)$ получился достаточно громоздким. Поэтому для несложных НАМ лучше использовать другой подход: прежде всего следует понять, какую задачу решает каждый из алгоритмов H_1 и H_2 , затем надо последовательность этих задач объединить в общую задачу и, наконец, построить алгоритм для решения этой общей задачи. Такой алгоритм и будет композицией исходных алгоритмов.

В нашем примере алгоритм H_1 заменяет на символ a только первое вхождение abb , а алгоритм H_2 удаляет только первое вхождение b . Значит, последовательное применение сначала H_1 , а затем H_2 приводит к тому, что сначала во входном слове самое левое вхождение abb заменяется на a , а затем в получившемся слове удаляется самый левый символ b . НАМ H' , решающий такую задачу, может быть, например, следующим:

$$H': \begin{cases} a^* \rightarrow *a & (1) \\ b^* \rightarrow *b & (2) \\ * \rightarrow \# & (3) \\ \#a \rightarrow a\# & (4) \\ \#b \mapsto & (5) \\ \# \mapsto & (6) \\ abb \rightarrow *a & (7) \\ \rightarrow \# & (8) \end{cases}$$

Проверим работу H' на том же входном слове $P = aabaabbbba$:

$$\begin{aligned} aabaabbbba & \xrightarrow{7} aaba*abba \xrightarrow{1} aab*aabba \xrightarrow{2} aa*baabba \xrightarrow{1} \\ & \rightarrow a*abaabba \xrightarrow{1} *aabaabba \xrightarrow{3} \#aabaabba \xrightarrow{4} a\#abaabba \xrightarrow{4} \\ & aa\#baabba \xrightarrow{5} aaaabba \end{aligned}$$

Нетрудно видеть, что H' эквивалентен алгоритму H (построенному по формальному плану) и, следовательно, также является композицией заданных алгоритмов H_1 и H_2 . Но алгоритм H' был получен неформально, как решение новой задачи: заменить в слове первое вхождение abb на символ a , после чего в получившемся слове удалить первое вхождение b .

3.5 Задачи для самостоятельного решения

3.1 Из указанного НАМ вычеркнуть ровно одну формулу подстановки так, чтобы получился алгоритм, применимый ко всем словам в алфавите $\{a,b\}$:

$$\begin{cases} a \rightarrow b \\ ba \rightarrow aba \\ b \rightarrow a \end{cases}$$

3.2 Определить область применимости указанного НАМ относительно алфавита $\{a,b,c\}$:

$$\text{а) } \begin{cases} a \rightarrow b \\ b \rightarrow c \\ c \rightarrow a \end{cases} \quad \text{б) } \begin{cases} a \rightarrow \\ bb \rightarrow b \\ ccc \rightarrow cc \end{cases} \quad \text{в) } \begin{cases} a \rightarrow \\ b \rightarrow b \\ c \rightarrow \end{cases} \quad \text{г) } \begin{cases} a \rightarrow c \\ b \rightarrow a \\ cc \rightarrow \\ c \rightarrow c \end{cases}$$

$$\begin{array}{l}
 \text{д)} \left\{ \begin{array}{l} ba \rightarrow ab \\ ca \rightarrow ac \\ cb \rightarrow bc \\ abc \vdash \\ \rightarrow \end{array} \right. \quad \text{е)} \left\{ \begin{array}{l} b \rightarrow a \\ a \rightarrow c \\ \rightarrow \\ c \rightarrow b \end{array} \right. \quad \text{ж)} \left\{ \begin{array}{l} a \rightarrow b \\ bb \rightarrow b \\ ccc \rightarrow ac \end{array} \right. \quad \text{з)} \left\{ \begin{array}{l} \rightarrow \\ \vdash \end{array} \right.
 \end{array}$$

3.3 Определить область применимости указанного НАМ относительно алфавита $\{a,b\}$:

$$\begin{array}{l}
 \text{а)} \left\{ \begin{array}{l} *ab \rightarrow ab \\ *a \rightarrow * \\ *b \rightarrow * \\ * \vdash \\ \rightarrow * \end{array} \right. \quad \text{б)} \left\{ \begin{array}{l} ab \rightarrow ba \\ ba \vdash ab \\ b \rightarrow bb \end{array} \right. \quad \text{в)} \left\{ \begin{array}{l} aa \rightarrow ba \\ b \rightarrow a \end{array} \right.
 \end{array}$$

3.4 Построить НАМ, который из всех слов в алфавите $\{a,b,c\}$ применим только к двум словам – пустому слову и слову $abccba$.

3.5 Построить НАМ, в котором не более 5 формул подстановки и который из всех слов в алфавите $\{a,b\}$ применим только к словам, имеющим следующий вид:

$$\begin{array}{ll}
 \text{а)} a^n b^n, \text{ где } n \geq 0 & \text{б)} a^n b^m, \text{ где } n \neq m, n \geq 0, m \geq 0 \\
 \text{в)} a^n b^m, \text{ где } n \geq m \geq 0 & \text{г)} a^n b^m, \text{ где } n > m \geq 0
 \end{array}$$

3.6 Построить НАМ, в котором не более 5 формул подстановки и который из всех слов в алфавите $\{a,b,c\}$ применим только к тем словам, длина которых:

$$\begin{array}{ll}
 \text{а)} \text{ кратна } 5 & \text{б)} \text{ не кратна } 5 \\
 \text{в)} \text{ больше } 4 & \text{г)} \text{ меньше } 4 \\
 \text{д)} \text{ равна } 3 & \text{е)} \text{ не равна } 3
 \end{array}$$

3.7 Построить НАМ, в котором не более 3 формул подстановки и который из всех слов в алфавите $\{a,b\}$ применим только к тем словам, для которых выполняется хотя бы одно из следующих двух условий: 1) в слове меньше трёх символов a ; 2) число символов b кратно 3.

3.8 Пусть в каждой формуле подстановки некоторого НАМ число символов в левой части строго больше числа символов в правой части. Доказать, что такой НАМ применим к любому слову.

3.9 Определить, самоприменим ли указанный НАМ:

$$\text{а) } \{ \rightarrow \quad \text{б) } \{ \mapsto \quad \text{в) } \begin{cases} a \rightarrow b \\ b \rightarrow a \end{cases} \quad \text{г) } \begin{cases} b \rightarrow a \\ ab \rightarrow ab \\ aa \mapsto \end{cases}$$

$$\text{д) } \begin{cases} b \rightarrow a \\ aa \rightarrow b \end{cases} \{ \mapsto \rightarrow \mapsto \quad \text{Здесь стрелка «с хвостиком» – символ алфавита}$$

3.10 $A=\{a,b\}$. Построить НАМ, в котором не более 4 формул подстановки и который:

- а) самоприменим и применим ко всем словам в алфавите A ;
- б) самоприменим, но неприменим ко всем словам в алфавите A ;
- в) самоприменим и применим только к какому-то одному слову в алфавите A ;
- г) самоприменим и неприменим только к какому-то одному слову в алфавите A ;
- д) несамоприменим и неприменим ко всем словам в алфавите A ;
- е) несамоприменим и применим только к какому-то одному слову в алфавите A ;
- ж) несамоприменим и неприменим только к какому-то одному слову в алфавите A .

3.11 $A=\{a,b\}$. Используя не более 3-х формул подстановки, построить НАМ, который:

- а) самоприменим и применим только к тем словам в алфавите A , которые содержат хотя бы один символ a ;
- б) самоприменим и неприменим только к тем словам в алфавите A , которые содержат хотя бы один символ a ;
- в) несамоприменим и применим только к тем словам в алфавите A , которые содержат хотя бы один символ a ;
- г) несамоприменим и неприменим только к тем словам в алфавите A , которые содержат хотя бы один символ a .

3.12 Пусть в некотором НАМ все формулы являются обычными и есть формула с пустой левой частью. Что можно сказать про область применимости этого НАМ? Может ли такой НАМ быть самоприменимым? Ответ обосновать.

3.13 Пусть в левых и правых частях всех формул подстановки некоторого НАМ используются только символы a . Установить, может ли такой НАМ быть:

а) самоприменимым и неприменимым ни к одному слову в алфавите $\{a\}$;

б) несамоприменимым и применимым ко всем словам в алфавите $\{a\}$.

3.14 Для каждой указанной пары алгоритмов H_1 и H_2 определить, эквивалентны ли они относительно алфавита $\{a,b\}$:

$$\text{а) } H_1: \left\{ \begin{array}{l} a \rightarrow a \end{array} \right. \quad H_2: \left\{ \begin{array}{l} *a \rightarrow aa \\ a \rightarrow *a \end{array} \right.$$

$$\text{б) } H_1: \left\{ \begin{array}{l} ba \rightarrow ab \\ ab \rightarrow \end{array} \right. \quad H_2: \left\{ \begin{array}{l} ab \rightarrow \\ ba \rightarrow \end{array} \right.$$

$$\text{в) } H_1: \left\{ \begin{array}{l} ab \rightarrow \end{array} \right. \quad H_2: \left\{ \begin{array}{l} *ab \rightarrow * \\ *a \rightarrow a* \\ *b \rightarrow b* \\ * \mapsto \\ \rightarrow * \end{array} \right.$$

$$\text{г) } H_1: \left\{ \begin{array}{l} ab \rightarrow \end{array} \right. \quad H_2: \left\{ \begin{array}{l} *ab \rightarrow \\ *a \rightarrow a* \\ *b \rightarrow b* \\ * \mapsto \\ \rightarrow * \end{array} \right.$$

$$\text{д) } H_1: \left\{ \begin{array}{l} *b \rightarrow b* \\ *a \mapsto \\ * \mapsto \\ a \rightarrow a* \end{array} \right. \quad H_2: \left\{ \begin{array}{l} *a \rightarrow a\# \\ *b \rightarrow b* \\ \#a \mapsto \\ \#b \rightarrow b* \\ \# \mapsto \\ \rightarrow * \end{array} \right.$$

3.15 Для указанного НАМ построить такой НАМ, который эквивалентен ему относительно алфавита $\{a,b\}$ и содержит только одну формулу подстановки:

$$\begin{array}{l}
 \text{a)} \left\{ \begin{array}{l} +a \rightarrow a+ \\ +b \rightarrow b+ \\ + \rightarrow * \\ a* \rightarrow *a \\ b* \rightarrow *b \\ * \mapsto a \\ \rightarrow + \end{array} \right. \quad
 \text{б)} \left\{ \begin{array}{l} *a \rightarrow b* \\ *b \rightarrow b* \\ * \mapsto \\ \rightarrow * \end{array} \right. \quad
 \text{в)} \left\{ \begin{array}{l} *a \rightarrow * \\ *b \rightarrow b* \\ * \rightarrow \\ \rightarrow * \end{array} \right.
 \end{array}$$

3.16 Из следующего НАМ вычеркнуть как можно больше формул так, чтобы получившийся алгоритм был эквивалентен ему относительно алфавита $\{a,b\}$:

$$\left\{ \begin{array}{l} aba \rightarrow aab \\ ba \rightarrow ab \\ abab \rightarrow aabb \\ ab \rightarrow \end{array} \right.$$

3.17 Выписать программу для машины Тьюринга, эквивалентную указанному НАМ относительно алфавита $\{a,b\}$:

$$\begin{array}{l}
 \text{a)} \left\{ \begin{array}{l} ab \mapsto ab \\ ba \mapsto ba \\ \rightarrow \end{array} \right. \quad
 \text{б)} \left\{ \begin{array}{l} aa \rightarrow aa \\ bb \rightarrow bb \end{array} \right. \quad
 \text{в)} \left\{ \begin{array}{l} *a \rightarrow b* \\ *b \mapsto b \\ \rightarrow * \end{array} \right.
 \end{array}$$

3.18 Даны следующие четыре НАМ:

$$\begin{array}{l}
 H_1: \left\{ \begin{array}{l} aa \rightarrow a \\ ab \mapsto abc \end{array} \right. \quad
 H_2: \left\{ \begin{array}{l} aa \rightarrow a \\ ab \rightarrow abd \end{array} \right. \quad
 H_3: \left\{ \begin{array}{l} aa \rightarrow a \\ ab \rightarrow ab \end{array} \right. \\
 H_4: \left\{ \begin{array}{l} aa \rightarrow a \\ ab \rightarrow abd \\ d \mapsto \end{array} \right.
 \end{array}$$

Какие из этих алгоритмов эквивалентны относительно алфавита $\{a,b\}$?

3.19 Для каждой пары алгоритмов H_1 и H_2 построить их композицию относительно алфавита $\{a,b\}$ в виде нормального алгоритма H ($H=H_2(H_1)$):

$$\begin{array}{l}
 \text{a)} H_1: \{ ba \rightarrow \} \quad H_2: \{ a \rightarrow bb \\
 \text{б)} H_1: \{ ab \rightarrow ba \} \quad H_2: \{ b \rightarrow a \}
 \end{array}$$

$$\text{в) } H_1: \begin{cases} *a \rightarrow b* \\ *b \rightarrow a* \\ * \mapsto \\ \rightarrow * \end{cases} \quad H_2: \{ ab \rightarrow \}$$

$$\text{г) } H_1: \begin{cases} abba \rightarrow abba \\ ab \rightarrow bbac \\ baa \mapsto \\ cba \rightarrow c \end{cases} \quad H_2: \begin{cases} c \rightarrow ab \\ bba \mapsto ab \end{cases}$$

*Рекомендация к (г):
Воспользоваться формальным способом из примера 6*

3.20 Заданы следующие три НАМ:

$$H_1: \{ ca \rightarrow ac \} \quad H_2: \{ cb \rightarrow bc \} \quad H: \begin{cases} ca \rightarrow ac \\ cb \rightarrow bc \end{cases}$$

Является ли алгоритм H композицией алгоритмов H_1 и H_2 ($H=H_2(H_1)$) относительно алфавита $\{a,b,c\}$? Ответ обосновать.

3.21 Имеются следующие пять НАМ:

$$H_1: \{ ab \rightarrow \} \quad H_2: \{ b \rightarrow a \} \quad H_3: \begin{cases} ab \rightarrow \\ b \rightarrow a \end{cases}$$

$$H_4: \begin{cases} ab \rightarrow \\ b \rightarrow a \\ \rightarrow * \\ * \mapsto \end{cases} \quad H_5: \begin{cases} *b \rightarrow a* \\ *a \rightarrow a* \\ * \mapsto \\ ab \rightarrow \\ \rightarrow * \end{cases}$$

Определить, есть ли среди этих алгоритмов такие, которые являются композицией двух других. Если да, то указать все такие композиции.

3.22 Имеется следующий НАМ в алфавите $\{a,b\}$

$$\{ a \rightarrow aa \}$$

Выписать программу для машины Тьюринга, эквивалентную данному НАМ. В машине Тьюринга должно быть не более двух состояний.

3.23 Пусть задан некоторый алфавит A . Может ли быть пустой область применимости (относительно алфавита A) самоприменимого алгоритма? Ответ обосновать.

3.24 Пусть задан некоторый алфавит A . Может ли быть непустой область применимости (относительно алфавита A) несамоприменимого алгоритма? Ответ обосновать.

3.25 Привести пример такого самоприменимого НАМ A_1 , в результате применения которого к своей записи получалась бы запись некоторого другого НАМ A_2 , такого что A_2 – несамоприменим.

3.26 $A = \{a, b\}$. Привести пример несамоприменимого нормального алгоритма в алфавите A , который применим только к словам, содержащим ровно три буквы a (например, к словам $babbaba$, $aaab$). Алгоритм должен содержать не более 6 формул подстановки.

3.27 Построить НАМ, эквивалентный заданному, и содержащий минимальное количество формул.

$$\left\{ \begin{array}{l} ** \rightarrow \#* \\ \#* \rightarrow \# \\ \#a \rightarrow a\# \\ \#b \rightarrow b\# \\ \# \mapsto \\ *aa \rightarrow a*a \\ *ab \rightarrow b*a \\ *bb \rightarrow b*b \\ *ba \rightarrow a*b \\ \rightarrow * \end{array} \right.$$

3.28 Привести пример такого самоприменимого НАМ, который был бы применим ко всем словам из алфавита $\{0,1\}$, за исключением слова 101. В записи алгоритма не должны использоваться дополнительные символы (спецзнаки).

3.29 Привести пример такого несамоприменимого НАМ в алфавите $\{0,1\}$, который был бы применим только к слову 101. В записи алгоритма не должны использоваться дополнительные символы (спецзнаки).

3.30 Введем следующие определения для скобочных систем. Скобочную систему будем называть *неделимой*, если ее нельзя представить в виде конкатенации (соединения) двух непустых скобочных систем. Например, системы $(())$ и $((()))$ неделимы, а системы $(()) (())$ и $(())$ не являются неделимыми.

Протяжение скобочной системы – это число неделимых скобочных систем, конкатенация которых дает данную систему. Протяжение пустой скобочной системы по определению равно нулю. Например, протяжение системы $(())$ равно 1, а протяжение $(()) (())$ равно 2.

Глубина скобочной системы – это максимальный уровень вложенности скобок. Глубина пустой системы по определению равна нулю. Например, глубина $((000)(0((0)0))$ равна 4, глубина $((000)0$ равна 2.

Ширина – максимальное протяжение среди всех подсистем данной скобочной системы. Например, ширина системы $((000)0$ равна 3.

Построить НАМ, который по заданной (правильной) скобочной системе вычисляет для нее:

- а) протяжение (существует решение в две формулы) ;
- б) глубину (существует решение в три формулы) ;
- в) ширину.

Ответ представить в единичной системе счисления (в виде палочек).

Рекомендуемая литература

1. Э.З. Любимский, В.В. Мартынюк, Н.П. Трифонов. Программирование. – М., “Наука”, 1980.

2. Л.С. Корухова, М.Р. Шура-Бура. Введение в алгоритмы (учебное пособие для студентов 1 курса, издание 2-е, исправленное). – М., Издательский отдел факультета ВМК МГУ, 2009.

3. А.А. Марков, Н.М. Нагорный. Теория алгорифмов. – М., ФАЗИС, 1996.

4. Ч. Петцольд. Читаем Тьюринга. Путешествие по исторической статье Тьюринга о вычислимости и машинах Тьюринга – М., ДМК Пресс, 2014.

Содержание

1. Машина Тьюринга	3
1.1 Краткое описание машины Тьюринга	3
1.2 Примеры на составление программ для МТ	8
1.3 Задачи для самостоятельного решения	20
2. Нормальные алгоритмы Маркова	24
2.1 Краткое описание нормальных алгоритмов Маркова	24
2.2 Примеры на составление НАМ	27
2.3 Задачи для самостоятельного решения	44
3. Задачи теоретического характера	51
3.1 Применимость алгоритма	52
3.2 Самоприменимость алгоритма	53
3.3 Эквивалентность алгоритмов	56
3.4 Композиция алгоритмов	58
3.5 Задачи для самостоятельного решения	64

Пильщиков Владимир Николаевич
Абрамов Владимир Геннадьевич
Вылиток Алексей Александрович
Горячая Илона Владимировна