Б. К. Мартыненко

ЯЗЫКИ *и* ТРАНСЛЯЦИИ

ИЗДАТЕЛЬСТВО САНКТ-ПЕТЕРБУРГСКОГО УНИВЕРСИТЕТА

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Б.К. Мартыненко

ЯЗЫКИ И ТРАНСЛЯЦИИ

Учебное пособие

ИЗДАТЕЛЬСТВО
С.-ПЕТЕРБУРГСКОГО УНИВЕРСИТЕТА
2004

УДК 519.685.3
ББК 32.81
М25

Рецензенты: д-р физ.-мат. наук проф. *И.Л. Братчиков* (С.-Петербургский гос. ун-т),
канд. физ.-мат. наук *Ю.Л. Костюк* (Томский гос. ун-т)

Печатается по постановлению
Редакционно-издательского совета
С.Петербургского государственного университета

Мартыненко Б.К.

М25 **Языки и трансляции:** Учеб. пособие — СПб.: Издательство С.-Петербургского университета, 2003. 235 с.
ISBN 5-288-02870-2

В учебном пособии излагаются основы математической теории формальных языков и трансляций, знание которой необходимо всем, кто работает в области теоретической или прикладной информатики. Излагаются основные факты теории с доказательствами. Содержание основано на материалах лекций, читавшихся автором в разные годы на математико-механическом факультете Ленинградского — С.-Петербургского государственного университета.

Прекрасные монографии по затрагиваемой тематике, изданные в 60–70-е годы, к настоящему времени стали библиографической редкостью. Предполагается, что настоящее пособие хотя бы частично компенсирует недостаток учебной литературы в данной области.

Пособие предназначено для студентов математических факультетов университетов, изучающих синтаксические методы в информатике.

Библиогр. 20 назв. Табл. 19. Ил. 30.

Тем. план 2002 г., № 64 ББК 32.81

ISBN 5-288-02870-2

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В современных информационных технологиях синтаксические методы играют существенную роль. Трансляторы языков программирования (компиляторы, интерпретаторы, конверторы), синтаксические редакторы, машинный перевод, различные средства обработки текстовой информации основаны на использовании синтаксических методов. Теория формальных языков и трансляций составляет теоретический фундамент этих методов. Основы этой теории были заложены Н. Хомским в 40–50-е годы XX столетия в связи с его лингвистическими работами, посвященными изучению естественных языков, но уже в следующем десятилетии синтаксические методы нашли широкое практическое применение в области разработки и реализации языков программирования. В настоящее время применение технологических средств, основанных на синтаксических методах, стало обыденным делом. Однако их грамотное и эффективное применение требует от пользователя знания, по крайней мере, основ математической теории, на которой они базируются.

Данное пособие написано на основе лекций, читавшихся автором в Ленинградском — Санкт-Петербургском государственном университете в разные годы для студентов математико-механического факультета. На отделении информатики этот курс читался совместно с С. Я. Фитиаловым — доцентом кафедры информатики, блестящим лектором, к сожалению, недавно ушедшим из жизни. Отчасти этим печальным обстоятельством вызвана подготовка данного пособия. Другая причина в том, что эта тематика, по-прежнему актуальная для специалистов и студентов, “вышла из моды”: фундаментальные монографии, изданные четверть века назад, стали библиографической редкостью. Автор надеется, что предлагаемое читателю пособие хотя бы в малой степени компенсирует существующую нехватку учебной литературы по изучаемому предмету.

Разбиение текста пособия на две части: I. Языки, грамматики, автоматы и II. Трансляции и синтаксические методы их реализации — соответствует былому распределению ролей двух лекторов, совместно читавших один курс. Часть I представляет собой авторизованный перевод отдельных глав монографии [2], с некоторыми дополнениями. Часть II является переложением нескольких тем из двухтомной монографии [6], отличающимся от оригинала, главным образом, тем, что в нем восстановлены полные доказательства многих утверждений, оставленных в первоисточнике в качестве упражнений. В Указателе литературы эти два источника следует считать основными.

Методические установки данного учебного пособия сводятся к тому, чтобы познакомить читателя с фундаментальными фактами классической теории языков и трансляций, дать ему представление о приемах доказательства утверждений в изучаемой области, а также снабдить его некоторым запасом практически полезных алгоритмов. Почти все доказательства носят конструктивный характер и потому дают полезный материал для практики.

Схематически содержание первой части следует в основном классификации языков по Н. Хомскому: в части I рассматриваются четыре типа грамматик и

соответствующих им распознавателей. Главу 6 о машинах Тьюринга в данном пособии можно рассматривать как факультативную, поскольку эта тема традиционно излагается в курсе по теории алгоритмов. Однако ее материал необходим для понимания последующих глав 7 и 8, в которых излагается связь языков типа 0 с машинами Тьюринга, и языков типа 1 — с линейно ограниченными автоматами. Несколько особняком стоит глава 9, в которой рассматриваются свойства замкнутости языков относительно элементарных операций и отображений. Она дополняет главу 3, в которой рассматриваются свойства замкнутости регулярных множеств.

Часть II посвящена описанию двух классов контекстно-свободных языков: $LL(k)$ и $LR(k)$ ввиду их практической важности. Первый является наибольшим естественным классом языков, допускающих детерминированный нисходящий разбор посредством k-предсказывающего алгоритма анализа. Второй представляет столь же естественный класс языков, допускающих детерминированный восходящий разбор посредством канонического анализатора Д. Кнута. Тот и другой имеют линейную сложность по времени. Как известно, метод Кнута используется в инструменте YACC — популярном построителе синтаксических анализаторов, а также в более развитых модификациях этого технологического средства. Фактически в них используются $LR(1)$-грамматики с дополнительными оптимизирующими ограничениями — так называемые $LALR(1)$-грамматики. Последние обсуждаются в конце этой части. В части II рассматриваются также некоторые классы синтаксически-управляемых трансляций с входными языками упомянутых классов и соответствующие алгоритмы их реализации. Последние описываются с достаточной для практики полнотой и обоснованием корректности и оценок сложности.

В Приложении приводится краткая сводка алгоритмически разрешимых и неразрешимых проблем, касающихся формальных языков. Она позволит практикующему читателю сориентироваться, что возможно или невозможно в мире формальных языков.

В пособии имеется достаточное число примеров, но нет задач и упражнений. Автор надеется исправить этот недостаток, подготавливая к изданию «Сборник задач и упражнений по теории формальных языков и трансляций».

Автор признателен рецензентам: профессору И. Л. Братчикову из Санкт-Петербургского государственного университета и доценту Ю. Л. Костюку из Томского государственного университета — за критические замечания, которые по возможности были приняты во внимание.

Особую благодарность хочется выразить заведующему лабораторией Java-технологии Научно-исследовательского института математики и механики СПбГУ им. академика В. И. Смирнова профессору В. О. Сафонову, предоставившему технические средства для подготовки оригинал-макета данного пособия.

Б. К. Мартыненко
Старый Петергоф,
Июнь 2002 г.

Глава 1

ЯЗЫКИ ИХ ПРЕДСТАВЛЕНИЕ

§ 1.1. Алфавиты и языки

Приступая к изучению теории языков, прежде всего следует определить, что мы подразумеваем под термином язык. Энциклопедическое определение языка как "важнейшего средства общения, обмена мыслями и взаимного понимания в человеческом обществе"[1] недостаточно точно для построения математической теории языков. Поэтому мы определим язык абстрактно — как математическую систему. Эта формальность даст нам возможность делать строгие утверждения о формальных языках, которые надлежащим образом будут моделироваться. С этой мыслью мы дадим следующие определения.

***Определение* 1.1.** *Алфавит* или *словарь* есть конечное множество символов.

Что такое *символ* — не определяется, как не определяется, например, точка в геометрии.

Некоторые примеры алфавитов: *латинский* $\{A, B, ..., Z\}$; *греческий* $\{\alpha, \beta, ..., \omega\}$; *бинарный* $\{0, 1\}$.

***Определение* 1.2.** *Предложение* (*строка*, *слово*) есть любая цепочка конечной длины, составленная из символов алфавита. Предложение, не содержащее ни одного символа, называется *пустым предложением*. Оно обозначается греческой буквой ε.

Если V — некоторый алфавит, то V^* обозначает множество всех предложений, составленных из символов алфавита V, включая и пустое предложение.

Будем обозначать посредством V^+ множество $V^* \setminus \{\varepsilon\}$. Так, например, если $V = \{0, 1\}$, то $V^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, ...\}$, а $V^+ = \{0, 1, 00, 01, 10, 11, ...\}$.

Если $x \in V^*$, то $|x|$ обозначает *длину* цепочки x, т. е. число символов, ее составляющих. Естественно считать, что длина пустой цепочки ε равна 0.

***Определение* 1.3.** *Язык* есть любое множество предложений над некоторым алфавитом. Формально, если V — *алфавит*, L — *язык*, то $L \subseteq V^*$.

Естественно возникают три вопроса:

1. Как представить язык, т. е. как определить предложения, составляющие язык?
2. Для каждого ли языка существует конечное представление?
3. Какова структура тех классов языков, для которых существуют конечные представления?

[1] Энциклопедический словарь. — М.: Большая советская энциклопедия, 1955. С. 716.

На первый вопрос ответить легко, если язык конечен. Надо просто перечислить все предложения, т. е. составить список. Если язык бесконечен, то возникает проблема нахождения способа *конечного представления* бесконечного языка. Это конечное представление само будет строкой символов над некоторым алфавитом с подразумеваемой интерпретацией, которая связывает это конкретное представление с данным языком.

На второй вопрос ответ отрицателен. В следующем параграфе будет показано, что множество всех предложений над данным словарем счетно-бесконечно.

То, что множество всех подмножеств счетно-бесконечного множества не является счетно-бесконечным, есть хорошо известный факт теории множеств[2]. Другими словами, множество всех языков над данным алфавитом не счетно. Ясно, что конечное представление языка будет предложением какого-то (обычно, другого) языка (*метаязыка*). Но этот последний, как и любой другой язык, не более чем счетно-бесконечное множество предложений. Так что существует значительно больше языков, чем конечных представлений.

Ответу на третий вопрос посвящена бóльшая часть дальнейшего изложения.

§ 1.2. Представление языков

Распознавание. Один из способов представления языка состоит в том, чтобы дать *алгоритм*[3], который определяет, есть ли данное предложение в данном языке или нет.

Более общий способ — дать *процедуру*, которая прекращает работу с ответом “да” для предложений в языке и либо не завершается[4], либо завершается с ответом “нет” для предложений не из языка. Говорят, что такие алгоритм и процедура *распознают* язык.

Существуют языки, которые можно распознать при помощи процедуры, но не при помощи алгоритма.

Порождение. Другой способ представления языка — дать процедуру, которая систематически порождает предложения языка последовательно в некотором порядке.

Если имеется алгоритм или процедура, которые распознают предложения языка над алфавитом V, то на их основе можно построить порождающий механизм — алгоритм или процедуру, порождающий этот язык. Именно, можно систематически генерировать все предложения из множества V^*, проверять каждое

---

[2] Теорема Г. Кантора (1878 г.).

[3] Согласно Д. Кнуту [11] *алгоритм* — это свод конечного числа правил, задающих последовательность выполнения операций при решении той или иной конкретной задачи. Для алгоритма характерны следующие пять особенностей: (1) *конечность* — завершаемость после выполнения конечного числа шагов; (2) *определенность* — однозначность; (3) *ввод* — исходные данные; (4) *вывод* — результат; (5) *эффективность* — выполнимость любой операции за конечное время. В вырожденных случаях ввод и (или) вывод могут отсутствовать.

[4] Этим она и отличается от алгоритма.

предложение на принадлежность его языку с помощью распознающего механизма и выводить в список только предложения языка. Однако если используется распознающая процедура, а не алгоритм, то есть опасность, что процесс порождения никогда не продвинется дальше первого же предложения, на котором процедура не завершается.

Для предотвращения этой опасности надо организовать проверку таким образом, чтобы распознающая процедура никогда не продолжала проверку одного предложения вечно. Сделать это можно следующим образом.

Пусть в алфавите V имеется p символов. Предложения из множества V^* можно рассматривать как числа p-ичной системы счисления плюс пустое предложение ε. Пронумеруем предложения из V^* в порядке увеличения длины, причем все предложения одинаковой длины будем нумеровать в "числовом" порядке. Например, если $V = \{a, b, c\}$, то предложения из V^* будут занумерованы следующим образом:

1.	ε	5.	*aa*	9.	*bb*
2.	*a*	6.	*ab*	10.	*bc*
3.	*b*	7.	*ac*	11.	*ca*
4.	*c*	8.	*ba*	12.	...

Тем самым показано, что множество предложений над алфавитом V счетно.

Пусть P — распознающая процедура для языка L. Предполагается, что она разбита на шаги так, что имеет смысл говорить об i-м шаге этой процедуры для любого заданного предложения.

Прежде чем дать процедуру перечисления предложений языка L, определим способ нумерации пар положительных целых. Можно отобразить все множество пар положительных целых (i, j) на множество положительных целых, как показано в табл. 1.1, при помощи следующей формулы:

$$k = \frac{(i + j - 1)(i + j - 2)}{2} + j.$$

Табл. 1.1

i \ j	1	2	3	4	5
1	1	3	6	10	15
2	2	5	9	14	...
3	4	8	13	...	
4	7	12	...		
5	11	...			

Замечание 1.1. Поясним, как получена вышеприведенная формула. Пусть имеется некоторое k, занимающее в сетке позицию (i, j). Ясно, что $k = N + j$, здесь N — число элементов внутри треугольной сетки с основанием, концы которого имеют координаты $(i, 1)$ и $(1, i)$. Очевидно, что

$$N = 1 + 2 + 3 + ... + n = \frac{n(n + 1)}{2},$$

здесь n — число рядов клеток треугольной сетки, параллельных ее основанию, считая и само это основание. Очевидно также, что $i+j=n+2$. Следовательно, $n=i+j-2$. Подставив последнее выражение для n в формулу для N, получим приведенное ранее выражение для k.

Итак, мы можем перенумеровать упорядоченные пары целых чисел согласно присвоенному номеру k. Несколько первых пар есть: (1, 1), (2, 1), (1, 2), (3, 1), (2, 2), ... Любая заданная пара чисел (i , j) действительно окажется в списке под номером

$$k = \frac{(i+j-1)(i+j-2)}{2} + j.$$

Теперь мы можем описать процедуру перечисления, т. е. *порождающую процедуру*, предложений языка L. Процедура перенумеровывает пары целых в соответствии с табл. 1.1. Когда пара (i , j) занумеровывается, процедура порождает i-е предложение из множества V^* и выполняет первые j шагов распознающей процедуры P над этим предложением. Когда процедура P определяет, что порожденное предложение принадлежит языку L, порождающая процедура добавляет это предложение к списку предложений языка L. Теперь, если слово номер i принадлежит языку L, то этот факт устанавливается при помощи процедуры P за j шагов для некоторого конечного j. Очевидно, что таким образом организованная процедура будет перечислять все предложения языка L.

Однако если имеется порождающая процедура для языка L, то на ее основе можно построить *распознающую процедуру* для этого языка. Действительно, чтобы определить, имеется ли предложение x в L, достаточно с помощью порождающей процедуры последовательно порождать предложения языка L и сравнивать каждое с x. Если x порождается, то распознающая процедура заканчивается, распознав x. Конечно, если предложение x не в L, то распознающая процедура никогда не закончится.

***Определение* 1.4.** *Я*зык, предложения которого могут порождаться процедурой, называется *рекурсивно перечислимым*. Также принято говорить, что язык рекурсивно перечислим, если существует процедура распознавания предложений этого языка.

***Определение* 1.5.** *Я*зык *рекурсивен*, если существует алгоритм его распознавания.

Известно, что класс рекурсивных языков является собственным подмножеством класса рекурсивно перечислимых языков. Более того, существуют языки и похуже, которые даже не рекурсивно перечислимы, т.е. невозможно эффективно перечислять предложения такого языка. Пример языка, не являющегося рекурсивно перечислимым множеством приводится в начале §7.3.

Теорема 1.1. *Пусть $L \subseteq V^*$ — некоторый язык, а $\overline{L} = V^* \setminus L$ — его дополнение. Если языки L и $\overline{L}$ рекурсивно перечислимы, то язык L рекурсивен.*

Доказательство. Пусть язык L распознается процедурой P, а его дополнение $\overline{L}$ распознается процедурой $\overline{P}$.

Построим алгоритм распознавания языка L, т.е. отвечающий на вопрос: $x \in L$?, следующим образом:

Шаг 1. $i := 1$.

Шаг 2. Применим i шагов процедуры P к цепочке x. Если процедура P не завершается, то перейти к шагу 3, иначе — к шагу 4.

Шаг 3. Применим i шагов процедуры $\overline{P}$ к цепочке x. Если процедура $\overline{P}$ не завершается, то $i := i + 1$ и перейти к шагу 2.

Шаг 4. При некотором i одна из этих двух процедур завершится, распознав цепочку x, так как либо $x \in L$ и это распознает процедура P, либо $x \notin L$ и это распознает процедура $\overline{P}$. Соответственно распознающий алгоритм выдает либо положительный, либо отрицательный ответ.

Поскольку язык L распознается описанным алгоритмом, то он *рекурсивен*. Что и требовалось доказать.

Завершая эту главу, можем сказать, что теория языков изучает множества строк символов (предложений), их представление, структуру и свойства.

Глава 2

ГРАММАТИКИ

§ 2.1. Мотивировка

Имеется один класс порождающих систем, которые представляют для нас первейший интерес — системы, называемые *грамматиками*.

Первоначально понятие грамматики было формализовано лингвистами при изучении естественных языков. Они интересовались не только определением, что является или не является правильным предложением языка, но также искали способы описания структуры предложений. Одной из целей была разработка формальной грамматики, способной описывать естественный язык. Надеялись, что, заложив в компьютер формальную грамматику, например, английского языка, можно сделать его "понимающим" этот язык, по словесной формулировке проблем получать их решения, осуществлять с помощью компьютера перевод с одного языка на другой.

Как показывает практика использования общедоступных программ машинного перевода, по-настоящему хорошего решения этой проблемы мы пока не имеем. Но вполне удовлетворительные результаты достигнуты в описании и реализации языков программирования.

Из школьного опыта известно, что представляет собой грамматический разбор предложения. При таком разборе определяется, какое слово является подлежащим, какое используется в роли сказуемого, какие слова играют роль определения, дополнения, обстоятельства и т.д. При разборе мы имеем дело с грамматическими категориями: предложение, группа существительного, группа сказуемого, существительное, глагол, наречие и т.д. и пользуемся собственно словами, составляющими разбираемое предложение. Например, структуру английского предложения: "The little boy run quickly" можно изобразить в виде диаграммы (рис. 2.1).

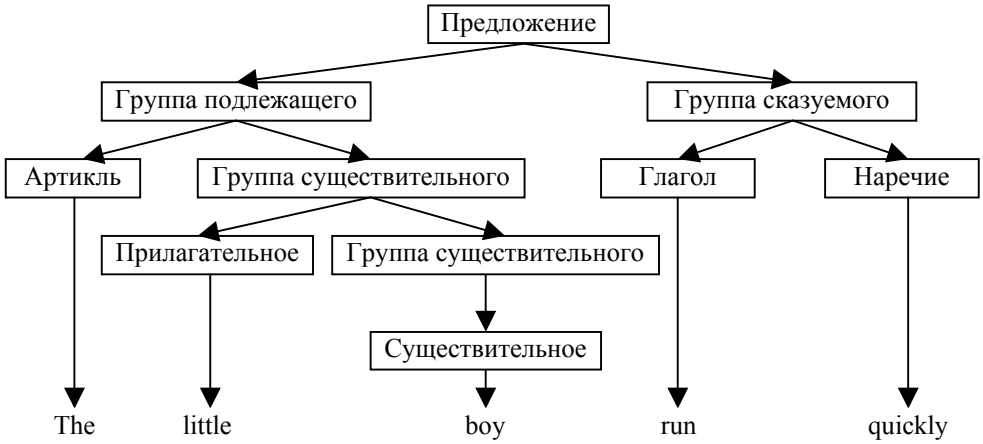

Рис. 2.1.

Грамматический разбор предложений подразумевает использование правил некоторой грамматики. Мы их будем представлять в следующей форме:

⟨ предложение ⟩ → ⟨ группа подлежащего ⟩ ⟨ группа сказуемого ⟩
⟨ группа подлежащего ⟩ → ⟨ артикль ⟩ ⟨ группа существительного ⟩
⟨ группа существительного ⟩→ ⟨ прилагательное ⟩ ⟨ группа существительного ⟩
⟨ группа существительного ⟩ → ⟨ существительное ⟩
⟨ группа сказуемого ⟩ → ⟨ глагол ⟩ ⟨ наречие ⟩
⟨ артикль ⟩ → The
⟨ прилагательное ⟩ → little
⟨ существительное ⟩ → boy
⟨ глагол ⟩ → run
⟨ наречие ⟩ → quickly

Здесь стрелочка отделяет левую часть правила от правой, а грамматические термины заключены в металингвистические скобки ⟨ и ⟩ для того, чтобы отличать их от слов, составляющих анализируемые предложения. По этим правилам можно не только проверять грамматическую правильность предложений, но также порождать грамматически правильные предложения. Механизм порождения прост. Начиная с грамматического термина ⟨ предложение ⟩, являющегося главным, каждый грамматический термин, замещается правой частью того правила, которое содержит его в левой части. Когда в результате таких замен в текущей цепочке не останется ни одного термина грамматики, мы получаем грамматически правильное предложение языка. Любое предложение языка, даже если он бесконечен, может быть выведено таким способом. И хотя большинство порождаемых предложений может не иметь смысла, тем не менее они считаются грамматически правильными.

Разумеется, приведенные здесь правила, не составляют грамматику настоящего английского языка, а служат лишь для иллюстрации разбора предложения, показанного на рис. 2.1.

§ 2.2. Формальное определение грамматики

В предыдущем параграфе речь шла о конкретной грамматике. Как видим, в ней имеются

1) грамматические термины (⟨ предложение ⟩, ⟨ группа подлежащего ⟩, ⟨ группа сказуемого ⟩ и т.д.), которые в теории формальных грамматик принято называть *нетерминалами;*

2) слова, составляющие предложения языка, они называются *терминалами*;

3) *правила*, левые и правые части которых состоят из нетерминалов и терминалов;

4) главный грамматический термин, называемый *начальным символом* или *начальным нетерминалом грамматики*, из него выводятся те цепочки терминалов, которые считаются предложениями языка (в рассмотренном в §2.1 примере начальным нетерминалом является символ ⟨ предложение ⟩).

***Определение* 2.1.** *Грамматикой* называется четверка $G = (V_N, V_T, P, S)$, где V_N, V_T — *алфавиты* (*словари*) *нетерминалов* и *терминалов* соответственно, причем $V_N \cap V_T = \varnothing$, P — конечное множество *правил*, каждое из которых имеет вид $\alpha \to \beta$, где $\alpha \in V^* V_N V^*$, $\beta \in V^*$, $V = V_N \cup V_T$ — *объединенный алфавит* (*словарь*) грамматики; S — *начальный нетерминал*.

В дальнейшем нетерминалы мы будем обозначать заглавными латинскими буквами, например S, A, B, C; терминалы — строчными буквами из начала латинского алфавита, например, a, b, c; цепочки терминалов — строчными буквами из конца латинского алфавита, например x, y, z; цепочки символов из объединенного алфавита — строчными греческими буквами, например α, β, γ.

Мы представили грамматику $G = (V_N, V_T, P, S)$, но не определили язык, который она порождает. Для этого нам потребуется ввести отношение непосредственной выводимости $\Rightarrow_G$, его транзитивное $\overset{+}{\Rightarrow}_G$ и рефлексивно-транзитивное замыкание $\overset{*}{\Rightarrow}_G$. Когда очевидно, в какой грамматике производится вывод, имя грамматики G можно не указывать.

***Определение* 2.2.** Пусть $\alpha \to \beta \in P$ — правило, γ, δ — любые цепочки из множества V^*. Тогда $\gamma\alpha\delta \Rightarrow_G \gamma\beta\delta$ читается следующим образом: “из $\gamma\alpha\delta$ *непосредственно выводится* $\gamma\beta\delta$ в грамматике G (при помощи данного правила)”.

Другими словами, символ $\Rightarrow_G$ связывает две цепочки тогда и только тогда, когда вторая цепочка получается из первой применением единственного правила.

***Определение* 2.3.** Пусть α_1, α_2, ..., α_m — цепочки из множества V^* и $\alpha_1 \Rightarrow_G \alpha_2$, $\alpha_2 \Rightarrow_G \alpha_3$, ..., $\alpha_{m-1} \Rightarrow_G \alpha_m$. Тогда мы пишем: $\alpha_1 \overset{*}{\Rightarrow}_G \alpha_m$ и говорим, что “из α_1 *выводится* α_m в грамматике G”.

Другими словами, $\alpha \overset{*}{\Rightarrow}_G \beta$, если β может быть получена из α путем применения некоторого числа правил из множества правил P. Считается, что $\alpha \overset{*}{\Rightarrow}_G \alpha$ для любой цепочки $\alpha \in V^*$ (рефлексивность) и для этого не требуется никаких правил.

Если мы хотим подчеркнуть, что такой вывод использует по крайней мере одно правило грамматики, то мы пишем: $\alpha \overset{+}{\Rightarrow}_G \beta$. Если мы хотим указать, что такой вывод происходит за n шагов, т.е. посредством применения n правил грамматики, то пишем: $\alpha \overset{n}{\Rightarrow}_G \beta$. Значок $\overset{n}{\Rightarrow}_G$ обозначает n-ю степень отношения непосредственной выводимости.

Напомним, что для любого отношения R имеют место следующие тождества:
$R^0 = E = \{(\alpha, \alpha) \mid \forall\, \alpha \in V^*\}$, $R^n = R R^{n-1} = R^{n-1} R$ для $n > 0$; в частности, $R^1 = R$;
$R^* = \bigcup_{k=0}^{k=\infty} R^k$, $R^+ = \bigcup_{k=1}^{k=\infty} R^k$.

Они, разумеется, применимы и к отношению непосредственной выводимости $\Rightarrow_G$.

***Определение* 2.4.** *Язык*, *порождаемый* грамматикой G, определим как

$$L(G) = \{w \mid w \in V_T^*,\ S \overset{*}{\Rightarrow}_G w\}.$$

Другими словами, язык есть множество терминальных цепочек, выводимых из начального нетерминала грамматики.

***Определение* 2.5.** Любая цепочка α, такая, что $\alpha \in V^*$ и $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$, называется *сентенциальной формой*.

***Определение* 2.6.** *Грамматики* G_1 *и* G_2 называются *эквивалентными,* если

$$L(G_1) = L(G_2).$$

Пример 2.1. Рассмотрим грамматику $G = (V_N, V_T, P, S)$, где $V_N = \{S\}$, $V_T = \{0, 1\}$, $P = \{S \to 0S1,\ S \to 01\}$. Здесь S — единственный нетерминал, он же — начальный; 0 и 1 — терминалы; правил два: $S \to 0S1$ и $S \to 01$.

Применив первое правило $n-1$ раз, а затем второе правило, получим:

$$S \underset{G}{\Rightarrow} 0S1 \underset{G}{\Rightarrow} 00S11 \underset{G}{\Rightarrow} 0^3S1^3 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} 0^{n-1}S1^{n-1} \underset{G}{\Rightarrow} 0^n1^n.$$

Здесь мы воспользовались обозначением $w^i = \underbrace{w \ldots w}_{i \text{ раз}}$, причем $w^0 = \varepsilon$.

Таким образом, эта грамматика порождает язык $L(G) = \{0^n1^n \mid n > 0\}$. Действительно, при использовании первого правила число символов S остается неизменным и равно 1. После использования второго правила число символов S в сентенциальной форме уменьшается на единицу. Поэтому после использования правила $S \to 01$ в результирующей цепочке не останется ни одного символа S. Так как оба правила имеют в левой части по символу S, то единственно возможный порядок их применения — сколько-то раз использовать первое правило, а затем один раз использовать второе.

Этот пример грамматики очень прост: было почти очевидно, какие предложения выводятся в ней. В общем случае определить, что порождается грамматикой, бывает очень трудно. Дадим более сложный пример.

Пример 2.2. Пусть $G = (V_N, V_T, P, S)$, где $V_N = \{S, B, C\}$, $V_T = \{a, b, c\}$, $P = \{(1)\ S \to aSBC,\ (2)\ S \to aBC,\ (3)\ CB \to BC,\ (4)\ aB \to ab,\ (5)\ bB \to bb,$
$(6)\ bC \to bc,\ (7)\ cC \to cc\}$.

Язык $L(G)$ содержит цепочки вида $a^nb^nc^n$ для каждого $n \geq 1$, так как мы можем использовать правило (1) $n-1$ раз, чтобы получить $S \overset{*}{\Rightarrow} a^{n-1}S(BC)^{n-1}$. Затем мы используем правило (2), чтобы получить $S \overset{*}{\Rightarrow} a^n(BC)^n$. Правило (3) дает нам возможность расположить B и C так, чтобы все B предшествовали всем C. Таким образом, $S \overset{*}{\Rightarrow} a^nB^nC^n$. Далее мы используем один раз правило (4) и получаем $S \overset{*}{\Rightarrow} a^nbB^{n-1}C^n$. Затем, используя правило (5) $n-1$ раз, получаем $S \overset{*}{\Rightarrow} a^nb^nC^n$. Применив один раз правило (6), получим $S \overset{*}{\Rightarrow} a^nb^ncC^{n-1}$. Наконец, применив $n-1$ раз правило (7), окончательно получим $S \overset{*}{\Rightarrow} a^nb^nc^n$.

Легко показать, что в общем случае возможен только такой порядок применения правил, т.е. что $L(G) = \{a^nb^nc^n \mid n \geq 1\}$.

§ 2.3. Типы грамматик

Грамматику, определенную в предыдущем параграфе, вслед за Н.Хомским назовем *грамматикой типа* 0. Им введено еще три типа грамматик, различающихся ограничениями, накладываемыми на вид правил.

***Определение* 2.7.** Грамматика $G = (V_N, V_T, P, S)$ является *грамматикой типа* 1 или *контекстно-зависимой грамматикой*, если для каждого ее правила $\alpha \to \beta \in P$ выполняется условие $|\beta| \geq |\alpha|$.

Часто вместо термина "контекстно-зависимая грамматика" используют аббревиатуру csg (context-sensitive grammar). Очевидно, что грамматики, приведенные в примерах 2.1. и 2.2, являются контекстно-зависимыми грамматиками, поскольку правые части их правил не короче левых частей.

Замечание 2.1. Некоторые авторы требуют, чтобы правила контекстно-зависимой грамматики имели вид: $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$, где $\alpha_1, \alpha_2, \beta \in V^*$, причем $\beta \neq \varepsilon$, а $A \in V_N$. Это мотивирует название "контекстно-зависимая", так как правило $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$ позволяет заменять A на β только, если A появляется в сентенциальной форме в контексте α_1 и α_2. В отечественной литературе для таких грамматик чаще используется термин НС-грамматики — *грамматики непосредственных составляющих*, а грамматики типа 1 называются *неукорачивающими грамматиками*.

Можно показать, что это требование не изменяет класса порождаемых языков. Действительно, любая НС-грамматика является неукорачивающей. Однако любое правило неукорачивающей грамматики может быть преобразовано так, чтобы все символы, его составляющие, были нетерминалами. Для этого достаточно каждое вхождение терминала $a \in V_T$ заменить на новый нетерминал Z, пополнить словарь нетерминалов этим символом и включить правило $Z \to a$ в множество правил грамматики. Правила вида $Z \to a$ допустимы для НС-грамматик.

Правило же вида

$X_1 X_2 \ldots X_m \to Y_1 Y_2 \ldots Y_{m+q}$, где $m > 0,\ q \geq 0,\ X_i, Y_j \in V_N,\ 1 \leq i \leq m,\ 1 \leq j \leq m + q$

эквивалентно группе правил:

$$
\begin{aligned}
&X_1 X_2 \ldots X_m \to A_1 X_2 \ldots X_m,\\
&A_1 X_2 \ldots X_m \to A_1 A_2 \ldots X_m,\\
&\qquad\qquad \ldots\\
&A_1 A_2 \ldots X_m \to A_1 A_2 \ldots A_m,\\
&A_1 A_2 \ldots A_m \to Y_1 A_2 \ldots A_m,\\
&Y_1 A_2 \ldots A_m \to Y_1 Y_2 \ldots A_m,\\
&\qquad\qquad \ldots\\
&Y_1 Y_2 \ldots A_{m-1} A_m \to Y_1 Y_2 \ldots Y_{m-1} A_m,\\
&Y_1 Y_2 \ldots Y_{m-1} A_m \to Y_1 Y_2 \ldots Y_{m-1} Y_m Y_{m+1} \ldots Y_{m+q},
\end{aligned}
$$

где $A_1, A_2, \ldots, A_m$ — дополнительные нетерминалы. Каждое из этих правил имеет требуемый НС-грамматикой вид: $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$.

***Определение* 2.8.** Грамматика $G = (V_N, V_T, P, S)$ является *грамматикой типа* 2 или *контекстно-свободной грамматикой*, если каждое ее правило имеет вид $A \to \beta \in P$, где $A \in V_N$, $\beta \in V^+$.

Вместо термина “контекстно-свободная грамматика” часто используют аббревиатуру cfg (context-free grammar) или сокращение КС-грамматика.

Замечание 2.2. Правило вида $A \to \beta$ позволяет заменить A на β независимо от контекста, в котором появляется A.

Грамматика, приведенная в примере 2.1, является контекстно-свободной.

Пример 2.3. Рассмотрим интересную контекстно-свободную грамматику $G = (V_N, V_T, P, S)$, где $V_N = \{S, A, B\}$, $V_T = \{a, b\}$, $P = \{S \to aB,\ S \to bA,\ A \to a,\ A \to aS,\ A \to bAA,\ B \to b,\ B \to bS,\ B \to aBB\}$.

Индукцией по длине цепочки покажем, что $L(G) = \{x \in \{a, b\}^+ \mid \#_a x = \#_b x\}$, где $\#_a x$ обозначает число букв a в цепочке x. Другими словами, язык, порождаемый этой грамматикой, состоит из непустых цепочек, в которых число букв a и b одинаково.

Достаточно доказать, что для $x \in V_T^+$

1) $S \overset{*}{\Rightarrow} x$ тогда и только тогда, когда x состоит из равного числа букв a и b;

2) $A \overset{*}{\Rightarrow} x$ тогда и только тогда, когда x имеет на одно a больше, чем b;

3) $B \overset{*}{\Rightarrow} x$ тогда и только тогда, когда x имеет на одно b больше, чем a.

База. Очевидно, что все три утверждения выполняются, если $|x| = 1$, поскольку $A \overset{*}{\Rightarrow} a$, $B \overset{*}{\Rightarrow} b$ и никакая терминальная цепочка не выводима из S. Кроме того, никакие строки единичной длины, отличающиеся от a и b, не выводимы из A и B соответственно.

Индукционная гипотеза. Предположим, что утверждения 1–3 выполняются для всех x, длина которых меньше k ($k \geq 1$).

Индукционный переход. Покажем, что они истинны для $|x| = k$.

Необходимость. Если $S \overset{*}{\Rightarrow} x$, то вывод должен начинаться либо с правила $S \to aB$, либо с правила $S \to bA$. В первом случае $x = ax_1$, причем $|x_1| = k - 1$ и $B \overset{*}{\Rightarrow} x_1$. По индукционному предположению число букв b в цепочке x_1 на единицу больше, чем число букв a, так что в цепочке x букв a столько же, сколько букв b. Аналогичное рассуждение достигает цели, если вывод начинается с правила $S \to bA$.

Достаточность. Теперь нужно доказать обратное утверждение, т.е. что если $|x| = k$ и $\#_a x = \#_b x$, то $S \overset{*}{\Rightarrow} x$. Либо первый символ x есть a, либо он есть b. Предположим, что $x = ax_1$. Теперь $|x_1| = k - 1$ и цепочка x_1 имеет на одну букву b больше, чем букв a. По индукционной гипотезе $B \overset{*}{\Rightarrow} x_1$. Но тогда

$$S \Rightarrow aB \overset{*}{\Rightarrow} ax_1 = x.$$

Аналогичное рассуждение достигает цели, если первый символ x есть b.

Для завершения доказательства аналогичным образом должны быть доказаны утверждения 2 и 3 (предоставим это читателю).

***Определение* 2.9.** Грамматика $G = (V_N, V_T, P, S)$ является *грамматикой типа* 3 или *регулярной грамматикой* (rg — regular grammar), если каждое ее правило имеет вид $A \to aB$ или $A \to a$, где $a \in V_T$; $A, B \in V_N$.

Замечание 2.3. В гл. 3 будет определено абстрактное устройство, называемое *конечным автоматом*, и показано, что языки, порождаемые грамматиками типа 3, являются в точности теми множествами, которые распознаются (допускаются) этими устройствами. Поэтому такой класс грамматик и языков часто называют *конечно-автоматными* или просто *автоматными*.

Пример 2.4. Рассмотрим грамматику $G = (V_N, V_T, P, S)$, где $V_N = \{S, A, B\}$, $V_T = \{0, 1\}$, $P = \{S \to 0A,\ S \to 1B,\ S \to 0,\ A \to 0A,\ A \to 0S,\ A \to 1B,\ B \to 1B,\ B \to 1, B \to 0\}$.

Ясно, что G — регулярная грамматика. Определить, какой язык порождает данная грамматика и доказать это, предоставляем читателю.

Очевидно, что каждая регулярная грамматика — контекстно-свободна; каждая контекстно-свободная грамматика — контекстно-зависима; каждая контекстно-зависимая грамматика является грамматикой типа 0. Каждому классу грамматик соответствует класс языков. Языку приписывается тип грамматики, которой он порождается. Например, контекстно-свободные грамматики (cfg) порождают контекстно-свободные языки (cfl), контекстно-зависимые грамматики (csg) порождают контекстно-зависимые языки (csl).

В соответствии с текущей практикой язык типа 3 или регулярный язык часто называют *регулярным множеством* (rs — regular set). Язык типа 0 называют *рекурсивно перечислимым множеством* (res — recursively enumerable set). Далее будет показано, что языки типа 0 соответствуют языкам, которые интуитивно могут быть перечислимы конечно описываемыми процедурами.

§ 2.4. Пустое предложение

Легко заметить, что по тому, как определены грамматики, пустое предложение (ε) не находится ни в контекстно-свободном (cfl), ни в контекстно-зависимом (csl) языках, ни в регулярном множестве (rs). Мы расширим данные ранее определения csg, cfg и rg, допустив порождение пустого предложения посредством правила вида $S \to \varepsilon$, где S — начальный символ, при условии, что S не появляется в правой части никакого правила. В этом случае ясно, что правило $S \to \varepsilon$ может использоваться только на первом шаге вывода и только на нем.

Имеет место следующая лемма.

Лемма 2.1. *Если $G = (V_N, V_T, P, S)$ есть контекстно-зависимая, контекстно-свободная или регулярная грамматика, то существует другая грамматика G_1 такого же типа, которая порождает тот же самый язык и в которой ни одно правило не содержит начальный символ в своей правой части.*

Доказательство. Пусть S_1 — символ, не принадлежащий ни алфавиту нетерминалов, ни алфавиту терминалов граммматики G. Положим $G_1 = (V_N \cup \{S_1\}, V_T, P_1, S_1)$. Множество правил P_1 состоит из всех правил P и правил вида $S_1 \to \alpha$ при условии, что $S \to \alpha \in P$. Поскольку символ $S_1 \notin V$ ($V = V_N \cup V_T$), то он не появляется в правой части никакого правила из множества P_1.

Докажем, что $L(G) = L(G_1)$.

I. Покажем сначала, что $L(G) \subseteq L(G_1)$.

Пусть $x \in L(G)$, т.е. $S \overset{*}{\underset{G}{\Rightarrow}} x$. Пусть первое используемое правило есть $S \to \alpha \in P$. Тогда $S \underset{G}{\Rightarrow} \alpha \overset{*}{\underset{G}{\Rightarrow}} x$. По построению P_1 правило $S_1 \to \alpha \in P_1$, так что $S_1 \underset{G_1}{\Rightarrow} \alpha$. Поскольку любое правило грамматики G является также правилом грамматики G_1, то $\alpha \overset{*}{\underset{G_1}{\Rightarrow}} x$. Таким образом, имеем $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x$, т.е. $x \in L(G_1)$ и тем самым доказано, что $L(G) \subseteq L(G_1)$.

II. Покажем теперь, что $L(G_1) \subseteq L(G)$.

Пусть $x \in L(G_1)$, т.е. $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x$. Пусть первое используемое правило есть $S_1 \to \alpha$. Но такое правило существует во множестве P_1 только потому, что в правилах P имеется правило $S \to \alpha$. Следовательно, $S \underset{G}{\Rightarrow} \alpha$. С другой стороны, $\alpha \overset{*}{\underset{G_1}{\Rightarrow}} x$ и α не содержит символа S_1. Поскольку ни одно правило из множества P_1 не содержит справа символа S_1, то ни одна сентенциальная форма этого вывода также не содержит символа S_1. Значит, в этом выводе используются только такие правила, которые имеются в множестве P. Поэтому $\alpha \overset{*}{\underset{G}{\Rightarrow}} x$. С учетом того, что $S \underset{G}{\Rightarrow} \alpha$, получаем вывод $S \underset{G}{\Rightarrow} \alpha \overset{*}{\underset{G}{\Rightarrow}} x$. Это и означает, что $L(G_1) \subseteq L(G)$.

Из утверждений I и II следует, что $L(G) = L(G_1)$.

Очевидно, что грамматики G и G_1 имеют один и тот же тип. Действительно, все правила грамматики G являются правилами грамматики G_1. Те же новые правила, которые имеются в грамматике G_1, но отсутствуют в грамматике G, отличаются от прототипа лишь символом слева, что не может изменить тип грамматики. Что и требовалось доказать.

Теорема 2.1. *Если L — контекстно-зависимый, контекстно-свободный или регулярный язык, то языки $L \cup \{\varepsilon\}$, $L \setminus \{\varepsilon\}$ также являются соответственно контекстно-зависимым, контекстно-свободным или регулярным языком.*

Доказательство. Согласно лемме 2.1 существует грамматика G, порождающая язык L, начальный нетерминал которой не встречается в правых частях ее правил.

Если язык $L = L(G)$ не содержит пустого предложения, то мы можем пополнить грамматику G еще одним правилом вида $S \to \varepsilon$. Обозначим пополненную грамматику G_1. Правило $S \to \varepsilon$ может использоваться только как первое и единственное правило вывода в G_1, поскольку начальный нетерминал S не встречается в правых частях правил. Любой вывод в грамматике G_1, не использующий правило $S \to \varepsilon$, есть также вывод в G, так что $L(G_1) = L(G) \cup \{\varepsilon\}$.

Если же $L = L(G)$ содержит пустое предложение, то среди правил грамматики G имеется правило вида $S \to \varepsilon$, с помощью которого только пустое предложение и выводится. Ни в каком другом выводе это правило не используется, так что, если его отбросить, то получим грамматику G_1, которая порождает все предложения языка L, кроме пустого. Следовательно, $L(G_1) = L(G) \setminus \{\varepsilon\}$.

Согласно лемме 2.1 типы грамматик G и G_1 одинаковы, поэтому одинаковы и типы языков, порождаемых этими грамматиками. Что и требовалось доказать.

Пример 2.5. Рассмотрим грамматику G из примера 2.2. Перестроим ее согласно лемме 2.1 так, чтобы начальный нетерминал не встречался в правых частях правил. Обозначим перестроенную грамматику G_1.

Очевидно, что $G_1 = (V_N, V_T, P_1, S_1)$, где $V_N = \{S_1, S, B, C\}$, $V_T = \{a, b, c\}$,

$P_1 = P \cup \{S_1 \to aSBC,\ S_1 \to aBC\} = \{S_1 \to aSBC,\ S_1 \to aBC$ (дополнительные правила),

1–7 — старые правила:

(1) $S \to aSBC$, (2) $S \to aBC$, (3) $CB \to BC$, (4) $aB \to ab$, (5) $bB \to bb$, (6) $bC \to bc$, (7) $cC \to cc$ }.

Построенная грамматика отличается от исходной грамматики G только дополнительным нетерминалом S_1, используемым в качестве нового начального символа, и двумя дополнительными правилами, его определяющими. Согласно лемме 2.1 $L(G_1) = L(G) = \{a^n b^n c^n \mid n \geq 1\}$.

Мы можем добавить пустое предложение к $L(G_1)$, пополнив грамматику G_1 еще одним правилом: $S_1 \to \varepsilon$. Обозначим пополненную грамматику G_2. Тогда $G_2 = (V_N, V_T, P_2, S_1)$, где $V_N = \{S_1, S, B, C\}$, $V_T = \{a, b, c\}$, $P_2 = P_1 \cup \{S_1 \to \varepsilon\}$.

Очевидно, что $L(G_2) = L(G_1) \cup \{\varepsilon\} = \{a^n b^n c^n \mid n \geq 0\}$.

§ 2.5. Рекурсивность контекстно-зависимых грамматик

Напомним, что грамматика $G = (V_N, V_T, P, S)$ — *рекурсивна*, если существует алгоритм, который определяет (за конечное время), порождается ли любая данная цепочка $x \in V_T^*$ данной грамматикой G.

Пусть $G = (V_N, V_T, P, S)$ — контекстно-зависимая грамматика. Проверить, порождается пустое предложение данной грамматикой или нет, просто. Достаточно посмотреть, имеется ли в ней правило $S \to \varepsilon$, поскольку $\varepsilon \in L(G)$ тогда и только тогда, когда $S \to \varepsilon \in P$. Если $\varepsilon \in L(G)$, то мы можем образовать новую грамматику: $G_1 = (V_N, V_T, P_1, S)$, отличающуюся от исходной лишь тем, что в ней не будет правила $S \to \varepsilon$, т.е. $P = P_1 \setminus \{S \to \varepsilon\}$. Согласно теореме 2.1 грамматика G_1 тоже контекстно-зависима, и $L(G_1) = L(G) \setminus \{\varepsilon\}$. Следовательно, в любом выводе длина последовательных сентенциальных форм разве лишь возрастает (т.е. не убывает).

Пусть объединенный словарь V грамматики G_1 имеет k символов. Предположим, что $S \overset{*}{\underset{G_1}{\Rightarrow}} x$, $x \neq \varepsilon$. Пусть этот вывод имеет вид: $S \underset{G_1}{\Rightarrow} \alpha_1 \underset{G_1}{\Rightarrow} \alpha_2 \underset{G_1}{\Rightarrow} \ldots \underset{G_1}{\Rightarrow} \alpha_m = x$, причем $|\alpha_1| \leq |\alpha_2| \leq \ldots \leq |\alpha_m|$. Предположим, что сентенциальные формы $\alpha_i, \alpha_{i+1}, \ldots, \alpha_{i+j}$ все одной и той же длины, скажем, p. Предположим также, что $j \geq k^p$. Тогда среди этих сентенциальных форм, по крайней мере, какие-то две одинаковы, так как число всевозможных различных непустых цепочек длиной p,

составленных из символов алфавита V, в котором k символов, равно k^p. В рассматриваемой же последовательности мы имеем $j + 1$ цепочек, где $j \geq k^p$. Пусть, например, $\alpha_q = \alpha_r$, где $q < r$. Тогда

$$S \underset{G_1}{\Rightarrow} \alpha_1 \underset{G_1}{\Rightarrow} \alpha_2 \underset{G_1}{\Rightarrow} \ldots \underset{G_1}{\Rightarrow} \alpha_q \underset{G_1}{\Rightarrow} \alpha_{r+1} \ldots \underset{G_1}{\Rightarrow} \alpha_m = x$$

является более коротким выводом цепочки x в грамматике G_1, чем первоначальный.

Интуитивно ясно, что если существует какой-нибудь вывод терминальной цепочки, то существует и "не слишком длинный" ее вывод.

В следующей теореме описывается алгоритм распознавания, в котором существенно используется это соображение.

Теорема 2.2. *Если грамматика* $G = (V_N, V_T, P, S)$ *— контекстно-зависима, то она — рекурсивна.*

Доказательство. В предыдущих параграфах было показано, что существует простой способ узнать, действительно ли $\varepsilon \in L(G)$, и если это так, то, исключив из грамматики правило $S \to \varepsilon$, получим грамматику, которая порождает тот же самый язык, но без пустого предложения. Любое же непустое предложение языка выводимо без использования этого правила. Поэтому, предполагая, что P не содержит правила $S \to \varepsilon$, рассмотрим произвольную цепочку $x \in V_T^+$. Наша задача найти алгоритм, разрешающий вопрос: $x \in L(G)$?

Пусть $|x| = n$ $(n > 0)$. Определим множество T_m следующим образом:

$$T_m = \{\alpha \mid S \overset{i}{\Rightarrow} \alpha,\ i \leq m,\ \alpha \in V^+,\ |\alpha| \leq n\}.$$

Другими словами, T_m содержит сентенциальные формы, выводимые не более, чем за m шагов, и не длиннее, чем n символов. Очевидно, что $T_0 = \{S\}$ и $T_m = T_{m-1} \cup \{\alpha \mid \beta \Rightarrow \alpha, \beta \in T_{m-1},\ |\alpha| \leq n\}$, т. е. T_m есть результат пополнения множества T_{m-1} цепочками, выводимыми из его цепочек за один шаг, длина которых не превосходит n.

Если $S \overset{*}{\Rightarrow} \alpha$ и $|\alpha| \leq n$, то $\alpha \in T_m$ при некотором m. Если $S \overset{*}{\Rightarrow} \alpha$ не имеет места или $|\alpha| > n$, то $\alpha \notin T_m$ ни при каком m. Также очевидно, что $T_{m-1} \subseteq T_m$ для всех $m \geq 1$. Поскольку T_m зависит только от T_{m-1}, то $T_m = T_{m+1} = T_{m+2} = \ldots$, если окажется на некотором шаге вычислений членов этой последовательности $T_m = T_{m-1}$.

Наш алгоритм будет вычислять T_1, T_2, T_3, ... до тех пор, пока для некоторого m не окажется $T_m = T_{m-1}$. Если цепочки x нет во множестве T_m, то ее нет и в $L(G)$, потому что $T_j = T_m$ для $j > m$. Конечно, если $x \in T_m$, то $S \overset{*}{\Rightarrow} x$.

Осталось доказать, что для некоторого m непременно будет $T_m = T_{m-1}$. Вспомним, что $T_i \supseteq T_{i-1}$ для каждого $i \geq 1$. При $T_i \neq T_{i-1}$ число элементов во множестве T_i, по крайней мере, на 1 больше, чем во множестве T_{i-1}. Если алфавит V имеет k символов, то число строк длиной n или меньше во множестве V^+ равно $k + k^2 + k^3 + \ldots + k^n \leq (k + 1)^n$. И это единственно возможные строки, которые могут быть в любом множестве T_i. Таким образом, при некотором $m \leq (k+1)^n$ непременно случится $T_m = T_{m-1}$. Следовательно, процесс вычисления

множеств $T_i\,(i>0)$ гарантированно закончится за конечное число шагов, и он тем самым является алгоритмом.

Замечание 2.4. Нет нужды доказывать, что алгоритм, описанный в теореме 2.2, применим также к контекстно-свободным и регулярным грамматикам.

Пример 2.6. Рассмотрим грамматику G из примера 2.2. С помощью только что описанного алгоритма проверим: $abac \in L(G)$? Получим, что

$T_0 = \{S\}$.
$T_1 = \{S, aSBC, aBC\}$.
$T_2 = \{S, aSBC, aBC, abC\}$.
$T_3 = \{S, aSBC, aBC, abC, abc\}$.
$T_4 = T_3$.

Поскольку $abac \notin T_3$, то $abac \notin L(G)$.

§ 2.6. Деревья вывода в контекстно-свободных грамматиках

Рассмотрим теперь наглядный метод описания любого вывода в контекстно-свободной грамматике. Фактически мы его уже могли наблюдать в §2.1.

***Определение* 2.10.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. Дерево есть *дерево вывода* для G, если оно удовлетворяет следующим четырем условиям:

1) каждый узел имеет метку — символ из алфавита V;
2) метка корня — S;
3) если узел имеет по крайней мере одного потомка, то его метка должна быть нетерминалом;
4) если узлы $n_1, n_2, \ldots, n_k$ — прямые потомки узла n, перечисленные слева направо, с метками $A_1, A_2, \ldots, A_k$ соответственно, а метка узла n есть A, то $A \to A_1A_2\ldots A_k \in P$.

Пример 2.7. Рассмотрим КС-грамматику $G = (\{S, A\}, \{a, b\}, P, S)$, где $P = \{S \to aAS, S \to a, A \to SbA, A \to ba, A \to SS\}$.

На рис. 2.2 изображено дерево, представляющее вывод:
$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$.

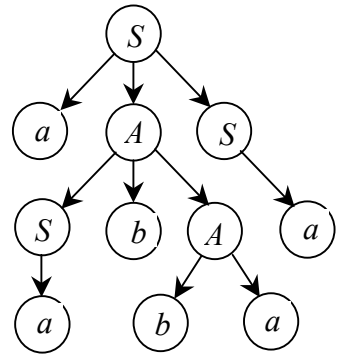

Рис. 2.2.

Результат *aabbaa* этого дерева вывода получается, если выписать метки листьев слева направо.

Имеет место следующая

Теорема 2.3. *Пусть* $G = (V_N, V_T, P, S)$ *— контекстно-свободная грамматика. Вывод* $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$, *где* $\alpha \in V^*$, $\alpha \neq \varepsilon$, *существует тогда и только тогда, когда существует дерево вывода в грамматике G с результатом* α.

Доказательство. Будем доказывать аналогичное утверждение для грамматик $G_A = (V_N, V_T, P, A)$ с одними и теми же V_N, V_T и P, но с разными начальными символами: $A \in V_N$. Если это вспомогательное утверждение будет доказано для любой грамматики G_A, то справедливость утверждения теоремы будет следовать просто как частный случай при $A = S$.

Поскольку, как было сказано, во всех грамматиках одни и те же правила, то утверждение $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$ эквивалентно утверждению $A \overset{*}{\underset{G_B}{\Rightarrow}} \alpha$ для любого $B \in V_N$, в частности при $B = S$, так как $G_S = G$, имеем также $A \overset{*}{\underset{G}{\Rightarrow}} \alpha$.

Все узлы дерева, не являющиеся листьями, будем называть *внутренними*.

I. Пусть $\alpha \in V^+$ есть результат дерева вывода для грамматики G_A. Индукцией по числу внутренних узлов k в дереве вывода покажем, что тогда $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$.

База. Пусть $k = 1$, тогда имеется только один внутренний узел. В этом случае дерево имеет такой вид, как показано на рис. 2.3.

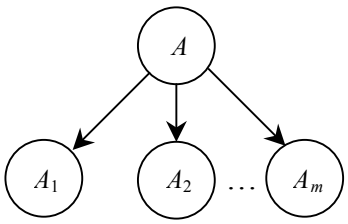

Рис. 2.3.

По определению дерева вывода $A \to A_1 A_2 \ldots A_m$ должно быть правилом грамматики G_A и, следовательно, вывод $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$ существует.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $k \le n$ ($n \ge 1$).

Индукционный переход. Пусть α есть результат дерева вывода с корнем, помеченным нетерминалом A, в котором k внутренних узлов, причем $k = n + 1$. Рассмотрим прямых потомков корня данного дерева вывода. Они не могут быть все листьями, так как в противном случае дерево имело бы только одну внутреннюю вершину — корень, а их должно быть не меньше двух.

Пусть метки прямых потомков корня, перечисленных в порядке слева направо: $A_1, A_2, \ldots, A_n$. Перенумеруем эти узлы в том же порядке: 1, 2, ... , n. По определению дерева вывода $A \to A_1 A_2 \ldots A_n \in P$.

Если узел i — не лист, то он — корень некоторого поддерева, в котором внутренних узлов не больше n. По индукционному предположению результат этого поддерева, обозначим его α_i, выводим из A_i, где A_i — нетерминал. В обо-

значениях это можно записать так: $A_i \overset{*}{\underset{G_{A_i}}{\Rightarrow}} \alpha_i$. Если же A_i — лист, то $A_i = \alpha_i$ и в этом случае $A_i \overset{*}{\underset{G_{A_i}}{\Rightarrow}} \alpha_i$.

Легко видеть, что если $i < j$, то узел i и все его потомки должны быть левее узла j и всех его потомков, и потому $\alpha = \alpha_1\alpha_2 \ldots \alpha_n$.

Мы можем теперь, используя правило $A \to A_1 A_2 \ldots A_n$ и все частичные выводы, выстроить вывод:

$A \underset{G_A}{\Rightarrow} A_1 A_2 \ldots A_n \overset{*}{\underset{G_A}{\Rightarrow}} \alpha_1 A_2 \ldots A_n \overset{*}{\underset{G_A}{\Rightarrow}} \alpha_1\alpha_2 \ldots A_n \overset{*}{\underset{G_A}{\Rightarrow}} \ldots \overset{*}{\underset{G_A}{\Rightarrow}} \alpha_1\alpha_2 \ldots \alpha_n = \alpha$.

Итак, $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$. Утверждение I доказано.

II. Пусть $A \overset{*}{\underset{G_A}{\Rightarrow}} \alpha$. Индукцией по длине вывода l покажем, что существует дерево вывода в грамматике G_A, результат которого есть α.

База. Пусть $l = 1$. Если $A \underset{G_A}{\Rightarrow} \alpha$, то на этом единственном шаге вывода используется правило $A \to \alpha \in P$. Если $\alpha = A_1 A_2 \ldots A_m$, то по определению дерево, показанное на рис. 2.3, есть дерево вывода в грамматике G_A. Очевидно, что его результат есть α.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \le n$ ($n \ge 1$).

Индукционный переход. Пусть $A \overset{l}{\underset{G_A}{\Rightarrow}} \alpha$, где $l = n + 1$. Этот вывод имеет длину, по крайней мере, 2. Следовательно, имеется первый шаг и n других шагов ($n \ge 1$), т. е. вывод имеет вид

$A \underset{G_A}{\Rightarrow} A_1 A_2 \ldots A_m \overset{l_1}{\underset{G_A}{\Rightarrow}} \alpha_1 A_2 \ldots A_m \overset{l_2}{\underset{G_A}{\Rightarrow}} \alpha_1\alpha_2 \ldots A_m \ldots \overset{l_m}{\underset{G_A}{\Rightarrow}} \alpha_1\alpha_2 \ldots \alpha_m = \alpha$.

Здесь $l_1 + l_2 + \ldots + l_m = n$, причем $l_i \le n$ ($1 \le i \le m$).

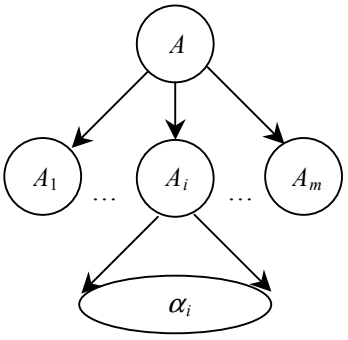

Рис. 2.4.

Если $l_i = 0$, то $\alpha_i = A_i$. Если $l_i > 0$, то по индукционному предположению существует дерево вывода T_i с корнем, имеющем метку A_i и результатом α_i. Но первый шаг вывода предполагает существование правила $A \to A_1 A_2 \ldots A_m \in P$. Следовательно, можно построить дерево вывода, верхняя часть которого будет иметь такой же вид, как на рис. 2.3.

Далее, те вершины, которые помечены символами A_i и для которых существуют выводы вида $A_i \overset{l_i}{\underset{G_A}{\Rightarrow}} \alpha_m$ при $l_i > 0$, заменим деревьями вывода T_i с корнями, помеченными A_i, и результатами α_i. То, что получится — см. рис. 2.4, является деревом вывода $A \overset{l}{\underset{G_A}{\Rightarrow}} \alpha$ в грамматике G_A. Утверждение II доказано.

Из I и II при $A = S$ следует утверждение теоремы.

Глава 3

КОНЕЧНЫЕ АВТОМАТЫ И РЕГУЛЯРНЫЕ ГРАММАТИКИ

§ 3.1. Конечный автомат

В гл. 2 мы познакомились со схемой порождения — грамматиками. Грамматики являются конечными описаниями языков. В этой главе мы рассмотрим другой метод конечного описания бесконечных языков — с помощью *распознавателей*, наипростейшим примером которых являются *конечные автоматы*.

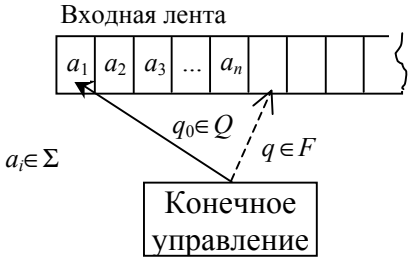

Рис.3.1.

Конечный автомат (рис. 3.1) состоит из конечного управления и входной ленты, разбитой на ячейки. В каждой ячейке записан один символ из входного алфавита Σ, и все они образуют конечную входную цепочку. Конечное управление первоначально находится в состоянии q_0 и сканирует крайнюю левую ячейку ленты. По мере чтения входной цепочки слева направо автомат переходит в другие состояния из множества Q. Если, прочитав входную цепочку, автомат оказывается в некотором конечном состоянии из множества F, то говорят, что он принял ее. Множество цепочек, принимаемых конечным автоматом, называется *языком*, *распознаваемым данным конечным автоматом*.

Конечные автоматы не могут распознавать все языки, порождаемые грамматиками, но языки, распознаваемые ими, являются в точности языками, порождаемыми грамматиками типа 3. В последующих главах мы познакомимся с распознавателями для языков типа 0, 1 и 2. В дальнейшем вместо термина “конечный автомат” будем использовать аббревиатуру fa — *finite automaton*.

Здесь мы определим конечный автомат как формальную систему и выясним его возможности как распознающего устройства.

***Определение* 3.1.** *Конечным автоматом* называется формальная система $M = (Q, \Sigma, \delta, q_0, F)$, где Q — конечное непустое множество *состояний*; Σ — конечный *входной алфавит*; δ — *отображение* типа $Q \times \Sigma \to Q$; $q_0 \in Q$ — *начальное состояние*; $F \subseteq Q$ — множество *конечных состояний*.

Запись $\delta(q, a) = p$, где $q, p \in Q$ и $a \in \Sigma$, означает, что конечный автомат M в состоянии q, сканируя входной символ a, продвигает свою входную головку на одну ячейку вправо и переходит в состояние p.

Область определения отображения δ можно расширить до $Q \times \Sigma^*$ следующим образом: $\delta'(q, \varepsilon) = q$, $\delta'(q, xa) = \delta(\delta'(q, x), a)$ для любого $x \in \Sigma^*$ и $a \in \Sigma$. Таким образом, запись $\delta'(q, x) = p$ означает, что fa M, начиная в состоянии $q \in Q$ чтение цепочки $x \in \Sigma^*$, записанной на входной ленте, оказывается в состоянии $p \in Q$, когда его входная головка продвинется правее цепочки x.

Далее мы будем использовать одно и то же обозначение δ для обоих отображений, так как это не приведет к путанице.

Определенная таким образом модель конечного автомата называется *детерминированной*. Для обозначения детерминированного автомата часто используют аббревиатуру dfa.

***Определение* 3.2.** Цепочка $x \in \Sigma^*$ *принимается конечным автоматом* M, если $\delta(q_0, x) = p$ для некоторого $p \in F$.

Множество всех цепочек $x \in \Sigma^*$, принимаемых конечным автоматом M, называется *языком*, *распознаваемым конечным автоматом* M, и обозначается как $T(M)$, т. е.

$$T(M) = \{x \in \Sigma^* \mid \delta(q_0, x) = p \text{ при некотором } p \in F\}.$$

Любое множество цепочек, принимаемых конечным автоматом, называется *регулярным*.

Пример 3.1. Рассмотрим диаграмму состояний конечного автомата. Пусть задан конечный автомат $M = (Q, \Sigma, \delta, q_0, F)$, где $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_0\}$, $\delta(q_0, 0) = q_2$, $\delta(q_0, 1) = q_1$, $\delta(q_1, 0) = q_3$, $\delta(q_1, 1) = q_0$,
$\delta(q_2, 0) = q_0$, $\delta(q_2, 1) = q_3$, $\delta(q_3, 0) = q_1$, $\delta(q_3, 1) = q_2$.

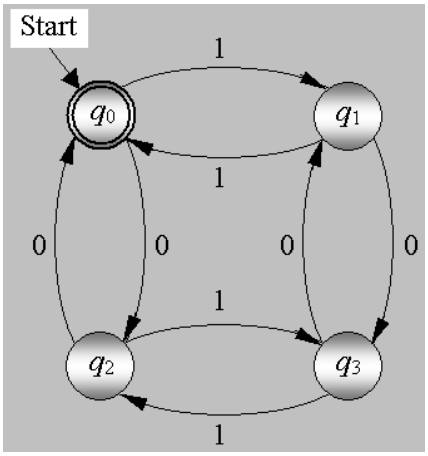

Рис. 3.2.

Диаграмма состояний конечного автомата состоит из узлов, представляющих состояния, и из ориентированных дуг, определяющих возможные переходы, которые зависят от входных символов. Так, если $\delta(q, a) = p$, то из узла, представляющего состояние q, в узел, представляющий состояние p, проводится дуга, помеченная входным символом a. На рис. 3.2 дана диаграмма состояний

конечного автомата M. Двойным кружком выделено единственное в данном примере конечное состояние, которое является одновременно и начальным.

Предположим, что на входе автомата находится цепочка 110101. Поскольку $\delta(q_0, 1) = q_1$, а $\delta(q_1, 1) = q_0$ и $q_0 \in F$, то цепочка 11 находится в языке, распознаваемом данным конечным автоматом, т.е. в $T(M)$, но мы интересуемся всей входной цепочкой. Сканируя остаток 0101 входной цепочки, автомат переходит последовательно в состояния q_2, q_3, q_1, q_0. Поэтому $\delta(q_0, 110101) = q_0$ и потому цепочка 110101 тоже находится в $T(M)$.

Легко показать, что $T(M)$ есть множество всех цепочек из $\{0, 1\}^*$, содержащих четное число нулей и четное число единиц. В частности, $\varepsilon \in T(M)$.

§ 3.2. Отношения эквивалентности и конечные автоматы

Напомним несколько понятий, относящихся к бинарным отношениям, которые потребуются в дальнейшем для описания свойств конечных автоматов.

***Определение* 3.3.** *Бинарное отношение* R на множестве S есть множество пар элементов S.

Если $(a, b) \in R$, то по-другому это записывают как aRb. Нас будут интересовать отношения на множествах цепочек над конечным алфавитом.

***Определение* 3.4.** Говорят, что бинарное отношение R на множестве S *рефлексивно*, если для каждого $s \in S$ имеет место sRs; *симметрично*, если для $s, t \in S$ sRt влечет tRs; *транзитивно*, если для $s, t, u \in S$ из sRt и tRu следует sRu.

Отношение, которое рефлексивно, симметрично и транзитивно, называется *отношением эквивалентности*.

Следующая теорема говорит о важном свойстве отношений эквивалентности.

Теорема 3.1. *Если R — отношение эквивалентности на множестве S, то S можно разбить на k непересекающихся подмножеств, называемых классами эквивалентности, так что aRb тогда и только тогда, когда a и b находятся в одном и том же подмножестве.*

Доказательство. Определим $[a]$ как $\{b \mid aRb\}$. Для любых a и b из множества S имеет место одно из двух соотношений: либо $[a] = [b]$, либо $[a] \cap [b] = \varnothing$. Начнем доказательство от противного.

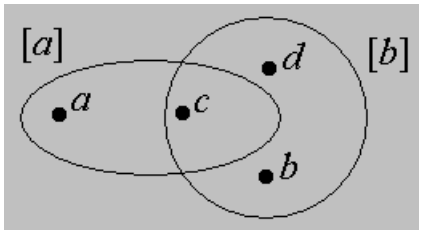

Рис. 3.3.

Пусть $[a] \neq [b]$ и $[a] \cap [b] \neq \varnothing$. Из $[a] \neq [b]$ следует, что существует $d \in S$, такое, что $d \notin [a]$ и $d \in [b]$ или (что то же самое) $d \,\overline{R}\, a$ и dRb.

Кроме того, из $[a] \cap [b] \neq \varnothing$ следует, что существует $c \in S$, такое, что $c \in [a]$ и $c \in [b]$ (рис. 3.3) или (что то же самое) cRa и cRb.

Итак, имеем dRb, cRb или по симметричности — bRc и, наконец, cRa. Из dRb, bRc и cRa по транзитивности получаем dRa. Но это противоречит предыдущему выводу: $d\overline{R}a$.

Отдельные множества $[a]$ для $a \in S$ являются *классами эквивалентности*. Ясно, что элементы a и b находятся в одном и том же множестве — классе эквивалентности тогда и только тогда, когда они эквивалентны, т.е. когда aRb.

***Определение* 3.5.** *Индекс отношения эквивалентности* R, заданного на множестве S, есть число образуемых им классов эквивалентности.

Замечание 3.1. Очевидно, что если S — конечно, то индекс отношения эквивалентности k не может быть бесконечным. В общем случае, когда S — бесконечно, k может быть как конечным, так и бесконечным.

Рассмотрим конечный автомат, приведенный в примере 3.1. Определим отношение R на множестве $\{0, 1\}^*$ следующим образом: $(x, y) \in R$ тогда и только тогда, когда $\delta(q_0, x) = \delta(q_0, y)$. Отношение R рефлексивно, симметрично и транзитивно, т.е. R — отношение эквивалентности. Отношение R делит множество $\{0, 1\}^*$ на четыре класса эквивалентности, соответствующие четырем состояниям автомата. Кроме того, если xRy, то для всех $z \in \{0,1\}^*$ имеет место $xzRyz$, поскольку $\delta(q_0, xz) = \delta(\delta(q_0, x), z) = \delta(\delta(q_0, y), z) = \delta(q_0, yz)$. Такое отношение эквивалентности называется *правоинвариантным*.

Теорема 3.2. *Следующие три утверждения эквивалентны*:

1) *язык* $L \subseteq \Sigma^*$ *принимается некоторым конечным автоматом*;

2) *язык* L *есть объединение некоторых классов эквивалентности правоинвариантного отношения эквивалентности конечного индекса*;

3) *пусть отношение эквивалентности* R *определяется следующим образом*: xRy *тогда и только тогда, когда для всех* $z \in \Sigma^*$ $xz \in L$ *точно тогда, когда* $yz \in L$. *Тогда* R *имеет конечный индекс.*

Доказательство.

1) $\rightarrow$ 2). Предположим, что язык L принимается некоторым конечным автоматом $M' = (Q', \Sigma', \delta', q_0', F')$. Пусть R' — отношение эквивалентности, определяемое следующим образом: $xR'y$ тогда и только тогда, когда $\delta'(q_0', x) = \delta'(q_0', y)$. Отношение R' правоинвариантно, поскольку, если $\delta'(q_0', x) = \delta'(q_0', y)$, то для любого $z \in \Sigma^*$ имеем $\delta'(q_0', xz) = \delta'(\delta'(q_0', x), z) = \delta'(\delta'(q_0', y), z) = \delta'(q_0', yz)$.

Индекс отношения R' конечен, поскольку (самое большее) он равен числу состояний fa M'. Кроме того, язык L есть объединение тех классов эквивалентности, которые включают элемент x, такой, что $\delta'(q_0', x) = p$, где $p \in F'$.

2) $\rightarrow$ 3). Покажем сначала, что любое отношение эквивалентности R', для которого выполняется утверждение 2, является уточнением отношения R, т.е.

каждый класс эквивалентности R' целиком содержится в некотором классе эквивалентности R. Если это так, то индекс отношения R не может быть больше индекса отношения R'. Индекс отношения R', как было показано, конечен. Следовательно, индекс отношения R тоже конечен.

Пусть $x R' y$. Так как отношение R' правоинвариантно, то для любого $z \in \Sigma^*$ имеет место $xz R' yz$ и, таким образом, $xz \in L$ точно тогда, когда $yz \in L$, т.е. $x R y$. Следовательно, $R' \subseteq R$, и потому $[x]_{R'} \subseteq [x]_R$. Это и значит, что любой класс эквивалентности отношения эквивалентности R' содержится в некотором классе эквивалентности отношения эквивалентности R.

Из этого следует, что индекс отношения эквивалентности R не может быть больше индекса отношения эквивалентности R'. Индекс отношения эквивалентности R' по предположению конечен. Следовательно, индекс отношения эквивалентности R тоже конечен.

3) $\rightarrow$ 1). Пусть $x R y$. Тогда для любых $w, z \in \Sigma^*$ цепочка $xwz \in L$ в точности тогда, когда цепочка $ywz \in L$. Следовательно, $xw R yw$, и потому R — правоинвариантно.

Построим конечный автомат $M = (Q, \Sigma, \delta, q_0, F)$, где в качестве Q возьмем конечное множество классов эквивалентности R, т.е. $Q = \{[x]_R \mid x \in \Sigma^*\}$; положим $\delta([x]_R, a) = [xa]_R$, и это определение непротиворечиво, так как R — правоинвариантно; положим $q_0 = [\varepsilon]$ и $F = \{[x]_R \mid x \in L\}$.

Очевидно, что конечный автомат M принимает язык L, поскольку $\delta(q_0, x) = \delta([\varepsilon]_R, x) = [x]_R$, и, таким образом, $x \in T(M)$ тогда и только тогда, когда $[x]_R \in F$.

Теорема 3.3. *Конечный автомат с минимальным числом состояний, принимающий язык L, единствен с точностью до изоморфизма* (*т.е. переименования состояний*) *и есть fa M из теоремы* 3.2.

Доказательство. При доказательстве теоремы 3.2 мы установили, что любой конечный автомат $M' = (Q', \Sigma', \delta', q_0', F')$, принимающий язык L, индуцирует отношение эквивалентности R', индекс которого не меньше индекса отношения эквивалентности R, определенного при формулировке утверждения 3 предыдущей теоремы. Поэтому число состояний fa M' больше или равно числу состояний fa M, построенного в третьей части доказательства теоремы 3.2.

Если M' — fa с минимальным числом состояний, то число его состояний равно числу состояний fa M и между состояниями M' и M можно установить одно-однозначное соответствие.

Действительно, пусть $q' \in Q'$. Должна существовать некоторая цепочка $x \in \Sigma^*$, такая, что $\delta'(q_0', x) = q'$, ибо в противном случае состояние q' без какого-нибудь ущерба для языка, принимаемого этим автоматом, можно было бы исключить из множества состояний Q' как недостижимое. Отбросив такое недостижимое состояние, мы получили бы автомат с меньшим числом состояний, ко-

торый принимал бы все тот же язык. Но это противоречило бы предположению, что M' является конечным автоматом с минимальным числом состояний.

Пусть $q' \in Q'$ и $q' = \delta'(q_0', x)$. Сопоставим с состоянием $q' \in Q'$ состояние $q \in Q$, достижимое автоматом M по прочтении той же цепочки x: $q = \delta(q_0, x) = \delta([\varepsilon]_R, x) = [x]_R$. Это сопоставление является непротиворечивым. Действительно, если q', $p' \in Q'$ и $q' = p'$, причем $q' = \delta'(q_0', x)$, а $p' = \delta'(q_0', y)$, то их образы есть соответственно $q = \delta(q_0, x) = \delta([\varepsilon]_R, x) = [x]_R$ и $p = \delta(q_0, y) = \delta([\varepsilon]_R, y) = [y]_R$. Учитывая, что x и y принадлежат одному и тому же классу эквивалентности отношения R' и что $R' \subseteq R$, заключаем, что x и y также находятся в одном и том же классе эквивалентности отношения R, т.е. $[x]_R = [y]_R$, и потому $q = p$. Другими словами, если прообразы состояний равны, то равны и их образы.

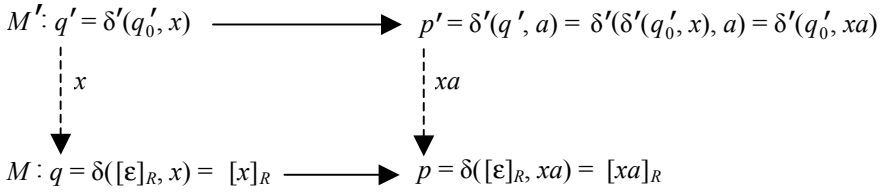

Рис. 3.4.

Кроме того, если fa M' совершает переход из состояния q' в состояние p', прочитав символ $a \in \Sigma$, то fa M переходит из состояния q, являющегося образом q', в состояние p, являющееся образом p', прочитав тот же самый символ $a \in \Sigma$, так как $\delta([x]_R, a) = [xa]_R$ (рис. 3.4).

§ 3.3. Недетерминированные конечные автоматы

Теперь мы введем понятие недетерминированного конечного автомата (ndfa — nondeterministic finite automaton). От своего детерминированного аналога он отличается только типом управляющего отображения (δ). Мы увидим, что любое множество, принимаемое недетерминированным конечным автоматом, может также приниматься детерминированным конечным автоматом. Но недетерминированный конечный автомат является полезным понятием при доказательстве теорем. Кроме того, с этого простейшего понятия легче начать знакомство с недетерминированными устройствами, которые не эквивалентны своим детерминированным аналогам.

***Определение* 3.6.** *Недетерминированным конечным автоматом* называется формальная система $M = (Q, \Sigma, \delta, q_0, F)$, где Q — конечное непустое множество *состояний*; Σ — *входной алфавит*; δ — *отображение* типа $Q \times \Sigma \to 2^Q$, $q_0 \in Q$ — *начальное состояние*; $F \subseteq Q$ — множество *конечных состояний*.

Существенная разница между детерминированной и недетерминированной моделями конечного автомата состоит в том, что значение $\delta(q, a)$ является (возможно пустым) множеством состояний, а не одним состоянием.

Запись $\delta(q, a) = \{p_1, p_2, \ldots, p_k\}$ означает, что недетерминированный конечный автомат M в состоянии q, сканируя символ a на входной ленте, продвигает входную головку вправо к следующей ячейке и выбирает любое из состояний $p_1, p_2, \ldots, p_k$ в качестве следующего.

Область определения δ может быть расширена на $Q \times \Sigma^*$ следующим образом:

$$\delta(q, \varepsilon) = \{q\},\ \delta(q, xa) = \bigcup_{p \in \delta(q, x)} \delta(p, a) \text{ для каждого } x \in \Sigma^* \text{ и } a \in \Sigma.$$

Область определения δ может быть расширена далее до $2^Q \times \Sigma^*$ следующим образом:

$$\delta(\{p_1, p_2, \ldots, p_k\}, x) = \bigcup_{i=1}^{k} \delta(p_i, x).$$

***Определение* 3.7.** Цепочка $x \in \Sigma^*$ *принимается недетерминированным конечным автоматом* M, если существует состояние p, такое, что $p \in F$ и $p \in \delta(q_0, x)$. Множество всех цепочек x, принимаемых ndfa M, обозначается $T(\mathrm{M})$.

Замечание 3.2. Напомним, что 2^Q, где Q — любое множество, обозначает степенное множество или множество всех подмножеств Q.

Пример 3.2. Рассмотрим недетерминированный конечный автомат, который распознает множество $\{0,1\}^*\{00,11\}\{0,1\}^*$:

$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_2, q_4\})$, где

$\delta(q_0, 0) = \{q_0, q_3\}$, $\delta(q_0, 1) = \{q_0, q_1\}$, $\delta(q_1, 0) = \varnothing$, $\delta(q_1, 1) = \{q_2\}$, $\delta(q_2, 0) = \{q_2\}$, $\delta(q_2, 1) = \{q_2\}$, $\delta(q_3, 0) = \{q_4\}$, $\delta(q_3, 1) = \varnothing$, $\delta(q_4, 0) = \{q_4\}$, $\delta(q_4, 1) = \{q_4\}$.

На рис. 3.5 приведена диаграмма состояний этого автомата. Фактически он принимает любые цепочки, составленные из нулей и единиц, в которых встречаются два подряд идущих нуля или единицы.

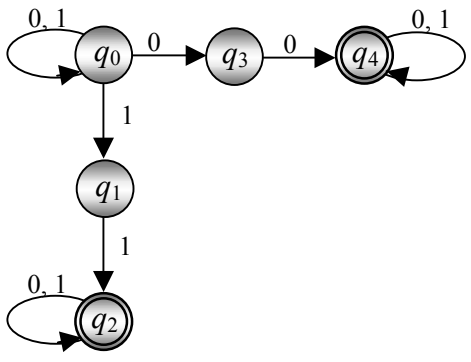

Рис. 3.5.

Теорема 3.4. *Пусть L — множество, принимаемое недетерминированным конечным автоматом. Тогда существует детерминированный конечный автомат, который принимает L.*

Доказательство. Пусть $M = (Q, \Sigma, \delta, q_0, F)$ — ndfa и $L = T(M)$. Определим dfa $M' = (Q', \Sigma, \delta', q_0', F')$ следующим образом. Положим $Q' = \{[s] \mid s \in 2^Q\}$. Состояние из множества Q' будем представлять в виде $[q_1, q_2, \ldots, q_i]$, где $q_1, q_2, \ldots, q_i$ — состояния из множества Q. Будем использовать обозначение φ для случая $i = 0$ или, что то же самое, $s = \varnothing$. Начальное состояние $q_0' = [q_0]$. Таким образом, dfa M' будет хранить след всех состояний, в которых ndfa M мог бы быть в любой данный момент. Пусть F' — множество всех состояний из Q', содержащих хотя бы одно состояние из множества конечных состояний F. Входной алфавит Σ — такой же, как в данном ndfa M.

Определим $\delta'([q_1, q_2, \ldots, q_i], a) = [p_1, p_2, \ldots, p_j]$ тогда и только тогда, когда $\delta(\{q_1, q_2, \ldots, q_i\}, a) = \{p_1, p_2, \ldots, p_j\}$.

Индукцией по длине l входной цепочки $x \in \Sigma^*$ легко показать, что $\delta'(q_0', x) = [q_1, q_2, \ldots, q_i]$ тогда и только тогда, когда $\delta(q_0, x) = \{q_1, q_2, \ldots, q_i\}$.

База. Пусть $l = 0$. Утверждение выполняется, ибо $\delta'(q_0', \varepsilon) = q_0' = [q_0]$ и $\delta(q_0, \varepsilon) = \{q_0\}$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \le n$ $(n \ge 0)$.

Индукционный переход. Докажем, что тогда утверждение выполняется и для $l = n + 1$.

Пусть $x = za$, где $z \in \Sigma^*$, $|z| = n$, $a \in \Sigma$. Тогда $\delta'(q_0', x) = \delta'(q_0', za) = \delta'(\delta'(q_0', z), a)$. По индукционному предположению $\delta'(q_0', z) = [p_1, p_2, \ldots, p_j]$ тогда и только тогда, когда $\delta(q_0, z) = \{p_1, p_2, \ldots, p_j\}$. В то же время по построению $\delta'([p_1, p_2, \ldots, p_j], a) = [q_1, q_2, \ldots, q_i]$ тогда и только тогда, когда $\delta(\{p_1, p_2, \ldots, p_j\}, a) = \{q_1, q_2, \ldots, q_i\}$. Таким образом, $\delta'(q_0', x) = \delta'(q_0', za) = \delta'([p_1, p_2, \ldots, p_j], a) = [q_1, q_2, \ldots, q_i]$ тогда и только тогда, когда $\delta(q_0, x) = \delta(q_0, za) = \delta(\{p_1, p_2, \ldots, p_j\}, a) = \{q_1, q_2, \ldots, q_i\}$. Чтобы закончить доказательство, остается добавить, что $\delta'(q_0', x) \in F'$ точно тогда, когда $\delta(q_0, x)$ содержит состояние из множества конечных состояний F. Следовательно, $T(M) = T(M')$. Что и требовалось доказать.

Поскольку детерминированные (dfa) и недетерминированные (ndfa) конечные автоматы распознают одни и те же множества, мы будем называть их общим термином *конечные автоматы* (fa) в тех случаях, когда это различие не существенно.

Пример 3.3. Пусть $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_1\})$ — ndfa, где $\delta(q_0, 0) = \{q_0, q_1\}$, $\delta(q_0, 1) = \{q_1\}$, $\delta(q_1, 0) = \varnothing$, $\delta(q_1, 1) = \{q_0, q_1\}$.

Построим детерминированный конечный автомат, эквивалентный данному. Положим $M' = (Q', \{0, 1\}, \delta', q_0', F')$. Согласно теореме 3.4 в качестве состоя-

ний детерминированного автомата следует взять все подмножества множества $\{q_0, q_1\}$, включая пустое, т.е. $Q' = \{\varphi, [q_0], [q_1], [q_0, q_1]\}$, причем $q_0' = [q_0]$.

Конечные состояния автомата M' представлены теми подмножествами, которые содержат конечные состояния данного автомата (в нашем случае: q_1), т.е. $F' = \{[q_1], [q_0, q_1]\}$.

Наконец, $\delta'([q_0], 0) = [q_0, q_1]$, $\delta'([q_0], 1) = [q_1]$, $\delta'([q_1], 0) = \varphi$, $\delta'([q_1], 1) = [q_0, q_1]$, $\delta'([q_0, q_1], 0) = [q_0, q_1]$, $\delta'([q_0, q_1], 1) = [q_0, q_1]$, $\delta'(\varphi, 0) = \varphi$, $\delta'(\varphi, 1) = \varphi$.

Поясним, что $\delta'([q_0, q_1], 0) = [q_0, q_1]$, так как $\delta(q_0, 0) = \{q_0, q_1\}$, $\delta(q_1, 0) = \varnothing$, и $\{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$. Аналогично, $\delta'([q_0, q_1], 1) = [q_0, q_1]$, ибо $\delta(q_0, 1) = \{q_1\}$, $\delta(q_1, 1) = \{q_0, q_1\}$ и $\{q_1\} \cup \{q_0, q_1\} = \{q_0, q_1\}$.

§ 3.4. Конечные автоматы и языки типа 3

Теперь мы возвращаемся к связи языков, порождаемых грамматиками типа 3, с множествами, которые принимаются конечными автоматами. Для удобства рассуждений введем понятие *конфигурации конечного автомата*.

***Определение* 3.8.** Пусть $M = (Q, \Sigma, \delta, q_0, F)$ — конечный автомат. *Конфигурацией* конечного автомата M назовем состояние управления в паре с непрочитанной частью входной цепочки.

Пусть (q, ax) — конфигурация fa M, где $q \in Q$, $a \in \Sigma$, $x \in \Sigma^*$, и пусть $p = \delta(q, a)$ в случае, если M — dfa, или $p \in \delta(q, a)$ в случае, когда M — ndfa. Тогда fa M может перейти из конфигурации (q, ax) в конфигурацию (p, x), и этот факт мы будем записывать как $(q, ax) \vdash (p, x)$. Далее символом $\overset{*}{\vdash}$ обозначается рефлексивно-транзитивное замыкание этого отношения на множестве конфигураций. Запись $(q_0, x) \overset{*}{\vdash} (p, \varepsilon)$, где $p \in F$, равнозначна записи $x \in T(M)$.

Теорема 3.5. *Пусть* $G = (V_N, V_T, P, S)$ *— грамматика типа* 3. *Тогда существует конечный автомат* $M = (Q, \Sigma, \delta, q_0, F)$, *такой, что* $T(M) = L(G)$.

Доказательство. Построим ndfa M, о котором идет речь. В качестве состояний возьмем нетерминалы грамматики и еще одно дополнительное состояние $A \notin V_N$. Итак, $Q = V_N \cup \{A\}$. Начальное состояние автомата M есть S. Если множество P содержит правило $S \to \varepsilon$, то $F = \{S, A\}$. В противном случае $F = \{A\}$. Напомним, что начальный нетерминал S не будет появляться в правых частях правил, если $S \to \varepsilon \in P$.

Включим A в $\delta(B, a)$, если $B \to a \in P$. Кроме того, в $\delta(B, a)$ включим все $C \in V_N$, такие, что $B \to aC \in P$. Положим $\delta(A, a) = \varnothing$ для каждого $a \in V_T$.

Построенный автомат M, принимая цепочку x, моделирует ее вывод в грамматике G. Требуется показать, что $T(M) = L(G)$.

I. Пусть $x = a_1 a_2 \ldots a_n$ и $x \in L(G)$, $n \geq 1$. Тогда существует вывод вида

$$S \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \ldots \Rightarrow a_1 a_2 \ldots a_{n-1} A_{n-1} \Rightarrow a_1 a_2 \ldots a_{n-1} a_n,$$

где $A_1, \ldots, A_{n-1} \in V_N$. Очевидно, что в нем используются следующие правила:

$$S \to a_1 A_1,\ A_1 \to a_2 A_2,\ \ldots,\ A_{n-2} \to a_{n-1} A_{n-1},\ A_{n-1} \to a_n \in P.$$

По построению δ

$$A_1 \in \delta(S, a_1),\ A_2 \in \delta(A_1, a_2),\ \ldots,\ A_{n-1} \in \delta(A_{n-2}, a_{n-1}),\ A \in \delta(A_{n-1}, a_n).$$

Следовательно, существует последовательность конфигураций

$$(S, a_1 a_2 \ldots a_n) \vdash (A_1, a_2 \ldots a_n) \vdash \ldots \vdash (A_{n-1}, a_n) \vdash (A, \varepsilon),$$

причем $A \in F$ и потому $x \in T(M)$. Если же $x = \varepsilon$, то $x \in L(G)$, и поскольку в этом случае $(S, \varepsilon) \vdash^* (S, \varepsilon)$, $S \in F$, то $x \in T(M)$.

II. Пусть теперь $x = a_1 a_2 \ldots a_n$ и $x \in T(M)$, $n \geq 1$. Тогда существует последовательность конфигураций вида

$$(S, a_1 a_2 \ldots a_n) \vdash (A_1, a_2 \ldots a_n) \vdash \ldots \vdash (A_{n-1}, a_n) \vdash (A, \varepsilon),$$

где $A \in F$. Очевидно, что

$$A_1 \in \delta(S, a_1),\ A_2 \in \delta(A_1, a_2),\ \ldots,\ A_{n-1} \in \delta(A_{n-2}, a_{n-1}),\ A \in \delta(A_{n-1}, a_n).$$

Но это возможно лишь при условии, что существуют правила

$$S \to a_1 A_1,\ A_1 \to a_2 A_2,\ \ldots,\ A_{n-2} \to a_{n-1} A_{n-1},\ A_{n-1} \to a_n \in P.$$

Используя их, легко построить вывод вида

$$S \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \ldots \Rightarrow a_1 a_2 \ldots a_{n-1} A_{n-1} \Rightarrow a_1 a_2 \ldots a_{n-1} a_n = x,$$

т. е. $x \in L(G)$.

Если же $x = \varepsilon$ и $x \in T(M)$, то $(S, \varepsilon) \vdash^* (S, \varepsilon)$ и $S \in F$. Но это возможно, если только существует правило $S \to \varepsilon \in P$. А тогда $S \Rightarrow \varepsilon$ и $x \in L(G)$. Что и требовалось доказать.

Теорема 3.6. *Пусть $M = (Q, \Sigma, \delta, q_0, F)$ — конечный автомат. Существует грамматика G типа 3, такая, что $L(G) = T(M)$.*

Доказательство. Без потери общности можно считать, что M — dfa. Построим грамматику $G = (V_N, V_T, P, S)$, положив $V_N = Q$, $V_T = \Sigma$, $S = q_0$, $P = \{q \to ap \mid \delta(q, a) = p\} \cup \{q \to a \mid \delta(q, a) = p \text{ и } p \in F\}$. Очевидно, что G — грамматика типа 3.

I. Пусть $x \in T(M)$ и $|x| > 0$. Покажем, что $x \in L(G)$.

Предположим, что $x = a_1 a_2 \ldots a_n$, $n > 0$. Существует последовательность конфигураций автомата M: $(q_0, a_1 a_2 \ldots a_n) \vdash (q_1, a_2 \ldots a_n) \vdash \ldots \vdash (q_{n-1}, a_n) \vdash (q_n, \varepsilon)$, причем $q_n \in F$. Соответственно $\delta(q_0, a_1) = q_1$, $\delta(q_1, a_2) = q_2, \ldots, \delta(q_{n-1}, a_n) = q_n$. По построению в множестве правил P существуют правила вида $q_i \to a_{i+1} q_{i+1}$ ($i = 0, 1, \ldots, n-1$) и правило $q_{n-1} \to a_n$. С их помощью можно построить вывод

$$q_0 \Rightarrow a_1 q_1 \Rightarrow a_1 a_2 q_2 \Rightarrow \ldots \Rightarrow a_1 a_2 \ldots a_{n-1} q_{n-1} \Rightarrow a_1 a_2 \ldots a_n.$$

А это значит, что $x \in L(G)$.

II. Пусть $x \in L(G)$ и $|x| > 0$. Покажем, что $x \in T(M)$.

Предположим, что $x = a_1 a_2 \ldots a_n$, $n > 0$. Существует вывод вида

$$q_0 \Rightarrow a_1 q_1 \Rightarrow a_1 a_2 q_2 \Rightarrow \ldots \Rightarrow a_1 a_2 \ldots a_{n-1} q_{n-1} \Rightarrow a_1 a_2 \ldots a_n.$$

Соответственно существуют правила $q_i \to a_{i+1}q_{i+1}$ ($i = 0, 1, \ldots, n-1$) и правило $q_{n-1} \to a_n$. Очевидно, что они обязаны своим существованием тому, что $\delta(q_i, a_{i+1}) = q_{i+1}$ ($i = 0, 1, \ldots, n-1$) и $q_n \in F$. А тогда существует последовательность конфигураций fa M вида

$$(q_0, a_1a_2 \ldots a_n) \vdash (q_1, a_2 \ldots a_n) \vdash \ldots \vdash (q_{n-1}, a_n) \vdash (q_n, \varepsilon),$$

причем $q_n \in F$. Это значит, что $x \in T(M)$.

Если $q_0 \notin F$, то $\varepsilon \notin T(M)$ и $L(G) = T(M)$. Если $q_0 \in F$, то $\varepsilon \in T(M)$. В этом случае $L(G) = T(M) \setminus \{\varepsilon\}$. По теореме 2.1 мы можем получить из G новую грамматику G_1 типа 3, такую, что $L(G_1) = L(G) \cup \{\varepsilon\} = T(M)$. Что и требовалось доказать.

Пример 3.4. Рассмотрим грамматику типа 3 $G = (\{S, B\}, \{0, 1\}, P, S)$, где $P = \{S \to 0B,\ B \to 0B,\ B \to 1S,\ B \to 0\}$. Мы можем построить ndfa $M = (\{S, B, A\}, \{0, 1\}, \delta, S, \{A\})$, где δ определяется следующим образом:

1) $\delta(S, 0) = \{B\}$, 2) $\delta(S, 1) = \varnothing$,
3) $\delta(B, 0) = \{B, A\}$, 4) $\delta(B, 1) = \{S\}$,
5) $\delta(A, 0) = \varnothing$, 6) $\delta(A, 1) = \varnothing$.

По теореме 3.5 $T(M) = L(G)$, в чем легко убедиться непосредственно.

Теперь мы используем построения теоремы 3.4, чтобы найти dfa M_1, эквивалентный автомату M.

Положим $M_1 = (Q_1, \{0, 1\}, \delta_1, [S], F_1)$, где $Q_1 = \{\varphi, [S], [B], [A], [S, B], [S, A], [B, A], [S, B, A]\}$, $F_1 = \{[A], [S, A], [B, A], [S, B, A]\}$; δ_1 определяется следующим образом:

1) $\delta_1([S], 0) = [B]$, 2) $\delta_1([S], 1) = \varphi$,
3) $\delta_1([B], 0) = [B, A]$, 4) $\delta_1([B], 1) = [S]$,
5) $\delta_1([B, A], 0) = [B, A]$, 6) $\delta_1([B, A], 1) = [S]$,
7) $\delta_1(\varphi, 0) = \varphi$, 8) $\delta_1(\varphi, 1) = \varphi$.

Имеются и другие определения δ_1. Однако ни в какие другие состояния, кроме φ, $[S]$, $[B]$, $[B, A]$ автомат M_1 никогда не входит, и все другие состояния и правила, определяющие и использующие их, могут быть удалены из множеств состояний Q_1, F_1 и δ_1 как бесполезные.

Теперь согласно построениям теоремы 3.6 по автомату M_1 построим грамматику типа 3:

$G_1 = (V_N, V_T, P, S)$, где $V_N = \{\varphi, [S], [B], [B, A]\}$, $V_T = \{0,1\}$, $S = [S]$,

$P = \{(1)\ [S] \to 0[B],\ (2)\ [S] \to 1\varphi,\ (3)\ [B] \to 0[B, A],\ (4)\ [B] \to 1[S],\ (5)\ [B] \to 0,$
$(6)\ [B, A] \to 0[B, A],\ (7)\ [B, A] \to 1[S],\ (8)\ [B, A] \to 0,\ (9)\ \varphi \to 0\varphi,$
$(10)\ \varphi \to 1\varphi\}$.

Грамматика G_1 значительно сложнее грамматики G, но $L(G_1) = L(G)$. Действительно, грамматику G_1 можно упростить, если заметить, что правила для $[B, A]$ порождают в точности те же цепочки, что и правила для $[B]$. Поэтому нетерминал $[B, A]$ можно заменить всюду на $[B]$ и исключить появившиеся дубликаты правил для $[B]$. Кроме того, отметим, что нетерминал φ не порождает ни

одной терминальной цепочки. Поэтому он может быть исключен вместе с правилами, в которые он входит. В результате получим грамматику

$G_2 = (\{[S], [B]\}, \{0, 1\}, P_2, [S])$, где $P_2 = \{(1)\ [S] \to 0[B],\ (2)\ [B] \to 0[B],$
$(3)\ [B] \to 1[S],\ (4)\ [B] \to 0\}$.

Очевидно, что полученная грамматика G_2 отличается от исходной грамматики G лишь обозначениями нетерминалов. Другими словами, $L(G_2) = L(G)$.

§ 3.5. Свойства языков типа 3

Поскольку класс языков, порождаемых грамматиками типа 3, равен классу множеств, принимаемых конечными автоматами, мы будем использовать обе формулировки при описании свойств класса языков типа 3. Прежде всего покажем, что языки типа 3 образуют *булеву алгебру*.

***Определение* 3.9.** *Булева алгебра* множеств есть совокупность множеств, замкнутая относительно объединения, дополнения и пересечения.

***Определение* 3.10.** Пусть $L \subseteq \Sigma_1^*$ — некоторый язык и $\Sigma_1 \subseteq \Sigma_2$. Под *дополнением* $\overline{L}$ языка L подразумевается множество $\Sigma_2^* \setminus L$.

Лемма 3.1. Класс языков типа 3 замкнут относительно объединения.

Доказательство. Возможны два подхода: один использует недетерминированные конечные автоматы, другой основывается на грамматиках. Мы будем пользоваться вторым подходом.

Пусть L_1 и L_2 — языки типа 3, порождаемые соответственно грамматиками типа 3: $G_1 = (V_N^{(1)}, V_T^{(1)}, P_1, S_1)$ и $G_2 = (V_N^{(2)}, V_T^{(2)}, P_2, S_2)$. Можно предположить, что $V_N^{(1)} \cap V_N^{(2)} = \varnothing$, ибо в противном случае этого всегда можно достичь путем переименования нетерминалов данных грамматик. Предположим также, что $S \notin (V_N^{(1)} \cup V_N^{(2)})$.

Построим новую грамматику $G_3 = \left(\{S\} \cup V_N^{(1)} \cup V_N^{(2)},\ V_T^{(1)} \cup V_T^{(2)},\ P_3,\ S\right)$, где $P_3 = (P_1 \cup P_2) \setminus \{S_1 \to \varepsilon,\ S_2 \to \varepsilon\} \cup \{S \to \alpha \mid \exists\ S_1 \to \alpha \in P_1$ или $\exists\ S_2 \to \alpha \in P_2\}$. Очевидно, что $S \underset{G_3}{\Rightarrow} \alpha$ тогда и только тогда, когда $S_1 \underset{G_1}{\Rightarrow} \alpha$ или $S_2 \underset{G_2}{\Rightarrow} \alpha$. В первом случае из α могут выводиться только цепочки в алфавите $V_N^{(1)} \cup V_T^{(1)}$, во втором — только цепочки в алфавите $V_N^{(2)} \cup V_T^{(2)}$. Формально, если $S_1 \underset{G_1}{\Rightarrow} \alpha$, то $\alpha \overset{*}{\underset{G_3}{\Rightarrow}} x$ тогда и только тогда, когда $\alpha \overset{*}{\underset{G_1}{\Rightarrow}} x$. Аналогично, если $S_2 \underset{G_2}{\Rightarrow} \alpha$, то $\alpha \overset{*}{\underset{G_3}{\Rightarrow}} x$ тогда и только тогда, когда $\alpha \overset{*}{\underset{G_2}{\Rightarrow}} x$. Сопоставив все сказанное, заключаем, что $S \overset{*}{\underset{G_3}{\Rightarrow}} x$ тогда и только тогда, когда либо $S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x$, либо $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} x$. А это и значит, что $L(G_3) = L(G_1) \cup L(G_2)$. Что и требовалось доказать.

Лемма 3.2. Класс множеств, принимаемых конечными автоматами (порождаемых грамматиками типа 3), замкнут относительно дополнения.

Доказательство. Пусть $M_1 = (Q, \Sigma_1, \delta_1, q_0, F)$ — dfa и $T(M_1) = S_1$. Пусть Σ_2 — конечный алфавит, содержащий Σ_1, и пусть $d \notin Q$ — новое состояние. Мы построим fa M_2, который принимает $\Sigma_2^* \setminus S_1$.

Положим $M_2 = (Q \cup \{d\}, \Sigma_2, \delta_2, q_0, (Q \setminus F) \cup \{d\})$, где (1) $\delta_2(q, a) = \delta_1(q, a)$ для каждого $q \in Q$ и $a \in \Sigma_1$, если $\delta_1(q, a)$ определено; (2) $\delta_2(q, a) = d$ для тех $q \in Q$ и $a \in \Sigma_2$, для которых $\delta_1(q, a)$ не определено; (3) $\delta_2(d, a) = d$ для каждого $a \in \Sigma_2$.

Интуитивно fa M_2 получается расширением входного алфавита fa M_1 до алфавита Σ_2, добавлением состояния "ловушки" d и затем перестановкой конечных и неконечных состояний. Очевидно, что fa M_2 принимает $\Sigma_2^* \setminus S_1$.

Теорема 3.7. *Класс множеств, принимаемых конечными автоматами, образует булеву алгебру.*

Доказательство непосредственно следует из лемм 3.1 и 3.2 и того факта, что $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Теорема 3.8. *Все конечные множества принимаются конечными автоматами.*

Доказательство. Рассмотрим множество, содержащее только одну непустую цепочку $x = a_1 a_2 \ldots a_n$. Мы можем построить конечный автомат M, принимающий только эту цепочку. Положим $M = (\{q_0, q_1, q_2, \ldots, q_n, p\}, \{a_1, a_2, \ldots, a_n\}, \delta, q_0, \{q_n\})$, где $\delta(q_i, a_{i+1}) = q_{i+1}$, $\delta(q_i, a) = p$, если $a \neq a_{i+1}$ $(i = 0, 1, \ldots, n-1)$, $\delta(q_n, a) = \delta(p, a) = p$ для всех a. Очевидно, что fa M принимает только цепочку x.

Множество, содержащее только пустую цепочку, принимается конечным автоматом $M = (\{q_0, p\}, \Sigma, \delta, q_0, \{q_0\})$, где $\delta(q_0, a) = \delta(p, a) = p$ для всех $a \in \Sigma$. Действительно, только пустая цепочка переведет автомат в состояние q_0, являющееся конечным.

Пустое множество принимается конечным автоматом $M = (\{q_0\}, \Sigma, \delta, q_0, \varnothing)$, где $\delta(q_0, a) = q_0$ для всех $a \in \Sigma$.

Утверждение теоремы немедленно следует из свойства замкнутости языков типа 3 относительно объединения. Что и требовалось доказать.

***Определение* 3.11.** *Произведением* или *конкатенацией* языков L_1 *и* L_2 называется множество $L_1L_2 = \{z \mid z = xy, x \in L_1, y \in L_2\}$. Другими словами, каждая цепочка в языке L_1L_2 есть конкатенация цепочки из L_1 с цепочкой из L_2.

Например, если $L_1 = \{01, 11\}$ и $L_2 = \{1, 0, 101\}$, то множество $L_1L_2 = \{011, 010, 01101, 111, 110, 11101\}$.

Теорема 3.9. *Класс множеств, принимаемых конечными автоматами, замкнут относительно произведения.*

Доказательство. Пусть $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ и $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ — детерминированные конечные автоматы, принимающие языки L_1 и L_2 соответственно.

Предположим, что $Q_1 \cap Q_2 = \varnothing$. Кроме того, без потери общности можно предположить, что $\Sigma_1 = \Sigma_2 = \Sigma$ (в противном случае мы могли бы добавить "мертвые" состояния в Q_1 и Q_2, как при доказательстве леммы 3.2).

Мы построим ndfa M_3, принимающий язык L_1L_2. Положим $M_3 = (Q_1 \cup Q_2, \Sigma, \delta_3, q_1, F)$, где

1) $\delta_3(q, a) = \{\delta_1(q, a)\}$ для любого $q \in Q_1 \setminus F_1$,

2) $\delta_3(q, a) = \{\delta_1(q, a), \delta_2(q_2, a)\}$ для любого $q \in F_1$,

3) $\delta_3(q, a) = \{\delta_2(q, a)\}$ для любого $q \in Q_2$.

Если $\varepsilon \notin L_2$, то $F = F_2$, иначе $F = F_1 \cup F_2$.

Правило 1 воспроизводит движения автомата M_1 до тех пор, пока он не достигает какого-нибудь из его конечных состояний, приняв некоторую (возможно пустую) начальную часть входной цепочки, принадлежащую языку L_1. Затем согласно правилу 2 он может продолжать повторять движения автомата M_1 или перейти в режим воспроизведения движений автомата M_2, начиная с его начального состояния. В последнем случае все дальнейшие движения благодаря правилу 3 повторяют движения автомата M_2. Если fa M_2 принимает (возможно пустое) окончание входной цепочки, принадлежащее языку L_2, то и автомат M_3 принимает всю входную цепочку. Другими словами, $T(M_3) = L_1L_2$.

***Определение* 3.12.** *Замыкание языка* L есть множество $L^* = \bigcup_{k=0}^{\infty} L^k$.

Предполагается, что $L^0 = \{\varepsilon\}$, $L^n = L^{n-1}L = LL^{n-1}$ при $n > 0$.

Пример 3.5. Если $L = \{01, 11\}$, то $L^* = \{\varepsilon, 01, 11, 0101, 0111, 1101, 1111, ...\}$.

Теорема 3.10. *Класс множеств, принимаемых конечными автоматами, замкнут относительно замыкания.*

Доказательство. Пусть $M = (Q, \Sigma, \delta, q_0, F)$ — dfa и $L = T(M)$. Построим ndfa M', который принимает язык L^*. Положим $M' = (Q \cup \{q_0'\}, \Sigma, \delta', q_0', F \cup \{q_0'\})$, где $q_0' \notin Q$ — новое состояние, и

$$\delta'(q_0', a) = \begin{cases} \{\delta(q_0, a), q_0\}, \text{ если } \delta(q_0, a) \in F, \\ \{\delta(q_0, a)\} \text{ в противном случае;} \end{cases}$$

$$\delta'(q, a) = \begin{cases} \{\delta(q, a), q_0\}, \text{ если } \delta(q, a) \in F, \\ \{\delta(q, a)\} \text{ в противном случае для всех } q \in Q. \end{cases}$$

Предназначение нового начального состояния q_0' — принимать пустую цепочку. Если $q_0 \notin F$, мы не можем просто сделать q_0 конечным состоянием, поскольку автомат M может снова прийти в состояние q_0, прочитав некоторую непустую цепочку, не принадлежащую языку L.

Докажем теперь, что $T(M') = L^*$.

I. Предположим, что $x \in L^*$. Тогда либо $x = \varepsilon$, либо $x = x_1x_2...x_n$, где $x_i \in L$ ($i = 1, 2, ..., n$). Очевидно, что автомат M' принимает ε. Ясно также, что из $x_i \in L$ следует, что $\delta(q_0, x_i) \in F$. Таким образом, множества $\delta'(q_0', x_i)$ и $\delta'(q_0, x_i)$ каждое со-

держит состояние q_0 и некоторое состояние p (возможно $p = q_0$) из множества F. Следовательно, множество $\delta'(q_0', x)$ содержит некоторое состояние из F и потому $x \in T(M')$.

II. Предположим теперь, что $x = a_1a_2 \ldots a_n \in T(M')$. Это значить, что

$$(q_0', a_1a_2 \ldots a_n) \vdash (q_1', a_2 \ldots a_n) \vdash \ldots \vdash (q_{n-1}', a_n) \vdash (q_n', \varepsilon),$$

причем $q_n' \in F \cup \{q_0'\}$. Ясно, что $q_n' = q_0'$ только в случае $n = 0$. В противном случае существует некоторая подпоследовательность состояний q_{i1}', $q_{i2}', \ldots, q_{im}'$ ($m \geq 1$) такая, что значение $q_{ik}' = q_0$ для всех $k = 1, 2, \ldots, m-1$, а $q_{im}' = q_n' \in F$. Это возможно только, если при некоторых j ($1 \leq j \leq n$) имеет место $q_j' \in \delta'(q_{j-1}', a_j)$ и $\delta(q_{j-1}', a_j) = q_j' \in F$. Поэтому $x = x_1x_2 \ldots x_m$, так что $\delta(q_0, x_k) \in F$ для $1 \leq k \leq m$. Это означает, что $x_k \in L$, а $x \in L^m \subset L^*$. Что и требовалось доказать.

Теорема 3.11. (С. Клини). *Класс множеств, принимаемых конечными автоматами, является наименьшим классом, содержащим все конечные множества, замкнутым относительно объединения, произведения и замыкания.*

Доказательство. Обозначим наименьший класс множеств, принимаемых конечными автоматами, содержащий все конечные множества и замкнутый относительно объединения, произведения и замыкания, через M. То, что класс множеств, принимаемых конечными автоматами, содержит класс M, является непосредственным следствием из леммы 3.1 и теорем 3.8–3.10. Остается показать, что класс M содержит класс множеств, принимаемых конечными автоматами.

Пусть L_1 — множество, принимаемое некоторым конечным автоматом $M = (\{q_1, q_2, \ldots, q_n\}, \Sigma, \delta, q_1, F)$. Пусть R_{ij}^k обозначает множество всех цепочек x, таких, что $\delta(q_i, x) = q_j$, причем, если y является непустым префиксом x, не совпадающим с x, то $\delta(q_i, y) = q_l$, где $l \leq k$. Другими словами, R_{ij}^k есть множество всех цепочек, которые переводят fa M из состояния q_i в состояние q_j, не проходя через какое-либо состояние q_l, где $l > k$. Заметим, что под “прохождением через состояние” мы подразумеваем вход и выход вместе. Но i и j могут быть больше k.

Мы можем определить R_{ij}^k рекурсивно:

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

$$R_{ij}^0 = \{a \mid a \in \Sigma, \delta(q_i, a) = q_j\}.$$

Приведенное определение R_{ij}^k неформально означает, что цепочки, которые переводят автомат M из состояния q_i в состояние q_j без перехода через состояния выше, чем q_k, (1) либо находятся во множестве R_{ij}^{k-1}, т.е. никогда не приводят автомат в состояние столь высокое, как q_k, (2) либо каждая такая цепочка состоит из цепочки во множестве R_{ik}^{k-1} (которая переводит автомат M в состояние q_k первый раз), за которой следует сколько-то цепочек из множества R_{kk}^{k-1}, переводящих автомат M из состояния q_k снова в состояние q_k без перехода через состояние q_k и высшие состояния, за которыми следует цепочка из множества

R_{kj}^{k-1}, переводящая автомат M из состояния q_k в состояние q_j, который при этом опять же не достигает состояния q_k и не проходит состояний с большими номерами.

Индукцией по параметру k мы можем показать, что множество R_{ij}^{k} для всех i и j находятся в пределах класса M.

База. Пусть $k = 0$. Утверждение очевидно, поскольку все множества R_{ij}^{0} являются конечными.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех k, таких, что $0 \le k \le m$ ($0 \le m < n$).

Индукционный переход. Докажем, что утверждение верно и для $k = m + 1$. Это так, поскольку R_{ij}^{m+1} выражается через объединение, конкатенацию и замыкание различных множеств вида R_{pq}^{m}, каждое из которых по индукционному предположению находится в классе M.

Остается заметить, что $L_1 = \bigcup\limits_{q_j \in F} R_{1j}^{n}$.

Таким образом, множество L_1 находится в классе M — наименьшем классе множеств, содержащем все конечные множества, замкнутом относительно объединения, конкатенации и замыкания. Что и требовалось доказать.

***Следствие* 3.1** (из теоремы Клини). Из теоремы 3.11 следует, что любое выражение, построенное из конечных подмножеств множества Σ^*, где Σ — конечный алфавит, и конечного числа операций объединения ‘$\cup$‘, произведения ‘.’ и замыкания ‘*’ со скобками, которые определяют порядок действий, обозначает множество, принимаемое некоторым конечным автоматом. И наоборот, каждое множество, принимаемое некоторым конечным автоматом, может быть представлено в виде такого выражения. Это обеспечивает нас хорошим средством для описания регулярных множеств. Оно называется *регулярным выражением*.

Пример 3.5: числа языка Паскаль. Пусть $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Тогда любое число языка Паскаль можно представить в виде следующего регулярного выражения:

$$D^{+}(\{.\}D^{+} \cup \{\varepsilon\})\,(\{e\}(\{+, -\} \cup \{\varepsilon\})\,D^{+} \cup \{\varepsilon\}).$$

Здесь использован символ *плюс Клини* ($^+$), определяемый следующими равенствами:

$$A^{+} = A^{*}A = A\,A^{*} = \bigcup_{k=1}^{\infty} A^{k}.$$

Напомним, что *звездочка Клини* (*), обозначающая замыкание, определяется следующим равенством:

$$A^{*} = \bigcup_{k=0}^{\infty} A^{k}.$$

Члены $\{\varepsilon\}$ обеспечивают необязательность дробной части и порядка (минимальная цепочка, представляющая число на языке Паскаль, состоит из одной цифры).

§ 3.6. Алгоритмически разрешимые проблемы, касающиеся конечных автоматов

В этом параграфе мы покажем, что существуют алгоритмы, отвечающие на многие вопросы, касающиеся конечных автоматов и языков типа 3.

Теорема 3.12. *Множество цепочек, принимаемых конечным автоматом с n состояниями*,

1) *не пусто тогда и только тогда, когда он принимает цепочку длиной, меньше n*;

2) *бесконечно тогда и только тогда, когда он принимает цепочку длиной* l, $n \le l < 2n$.

Доказательство.

Необходимость условия 1 вытекает из следующих рассуждений от противного. Предположим, что множество $T(M) \neq \varnothing$, но ни одной цепочки длиной меньше n в этом множестве не существует. Пусть $x \in T(M)$, где $M = (Q, \Sigma, \delta, q_0, F)$ — конечный автомат с n состояниями, и $|x| \ge n$. Пусть x — одна из самых коротких таких цепочек. Очевидно, что существует такое состояние $q \in Q$, что $x = x_1x_2x_3$, где $x_2 \neq \varepsilon$, и $\delta(q_0, x_1) = q$, $\delta(q, x_2) = q$, $\delta(q, x_3) \in F$. Но тогда $x_1x_3 \in T(M)$, поскольку $\delta(q_0, x_1x_3) = \delta(q_0, x_1x_2x_3) \in F$. В то же время $|x_1x_3| < |x_1x_2x_3| = |x|$, но это противоречит предположению, что x — одна из самых коротких цепочек, принимаемых fa M.

Достаточность условия 1 очевидна. Действительно, если конечный автомат принимает цепочку с меньшей длиной, чем n, то множество $T(M)$ уже не пусто (какой бы длины цепочка ни была).

Докажем теперь утверждение 2.

Необходимость условия 2 доказывается способом от противного. Пусть fa M принимает бесконечное множество цепочек, и ни одна из них не имеет длину l, $n \le l < 2n$.

Если бы в множестве $T(M)$ существовали только цепочки длиной $l < n$, то по доказанному язык был бы конечен, но это не так. Поэтому существуют и цепочки длиной $l \ge 2n$. Пусть x — одна из самых коротких цепочек, таких, что $x \in T(M)$ и $|x| \ge 2n$. Очевидно, что существует такое состояние $q \in Q$, что $x = x_1x_2x_3$, где $1 \le |x_2| \le n$, и $\delta(q_0, x_1) = q$, $\delta(q, x_2) = q$, $\delta(q, x_3) \in F$. Но тогда $x_1x_3 \in T(M)$, поскольку $\delta(q_0, x_1x_3) = \delta(q_0, x_1x_2x_3) \in F$ при том, что $|x_1x_3| \ge n$ (ибо $|x| = |x_1x_2x_3| \ge 2n$ и $1 \le |x_2| \le n$). Поскольку по предположению в $T(M)$ цепочек длиной $n \le l < 2n$ не существует, то $|x_1x_3| \ge 2n$. Следовательно, вопреки предположению, что $x = x_1x_2x_3 \in T(M)$ — одна из самых коротких цепочек, длина которой больше или равна $2n$, нашлась более короткая цепочка $x_1x_3 \in T(M)$ и тоже с длиной, большей или равной $2n$. Это противоречие доказывает необходимость условия 2.

Достаточность условия 2 вытекает из следующих рассуждений. Пусть существует $x \in T(M)$, причем $n \le |x| < 2n$. Как и ранее, можем утверждать, что су-

ществует $q \in Q$, $x = x_1x_2x_3$, где $x_2 \neq \varepsilon$, и $\delta(q_0, x_1) = q$, $\delta(q, x_2) = q$, $\delta(q, x_3) \in F$. Но тогда цепочки вида $x_1x_2^ix_3 \in T(M)$ при любом i. Очевидно, что множество $T(M)$ бесконечно. Что и требовалось доказать.

***Следствие* 3.2.** Из доказанной теоремы следует существование алгоритмов, разрешающих вопрос о пустоте, конечности и бесконечности языка, принимаемого любым данным конечным автоматом.

Действительно, алгоритм, проверяющий непустоту языка, может систематически генерировать все цепочки с постепенно увеличивающейся длиной, но меньшей n. Каждая из этих цепочек пропускается через автомат. Либо автомат примет какую-нибудь из этих цепочек, и тогда алгоритм завершится с положительным ответом, либо ни одна из этих цепочек не будет принята, и тогда алгоритм завершится с отрицательным результатом. В любом случае процесс завершается за конечное время.

Алгоритм для проверки бесконечности языка можно построить аналогичным образом, только он должен генерировать и тестировать цепочки длиной от n до $2n - 1$ включительно.

Теорема 3.13. *Существует алгоритм для определения, являются ли два конечных автомата эквивалентными* (*т.е. принимают ли они один и тот же язык*).

Доказательство. Пусть M_1 и M_2 — конечные автоматы, принимающие языки L_1 и L_2 соответственно. По теореме 3.7 множество $(L_1 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_2)$ принимается некоторым конечным автоматом M_3. Легко видеть, что множество $T(M_3)$ не пусто тогда и только тогда, когда $L_1 \neq L_2$. Следовательно, согласно теореме 3.12 существует алгоритм для определения, имеет ли место $L_1 = L_2$. Что и требовалось доказать.

Глава 4

КОНТЕКСТНО-СВОБОДНЫЕ ГРАММАТИКИ

§ 4.1. Упрощение контекстно-свободных грамматик

В этой главе мы опишем некоторые основные упрощения КС-грамматик и докажем несколько важных теорем о нормальных формах Хомского и Грейбах. Мы также покажем, что существуют алгоритмы для определения, является ли язык, порождаемый КС-грамматикой, пустым, конечным или бесконечным.

Будет определено так называемое свойство самовложенности КС-грамматик и показано, что КС-язык нерегулярен тогда и только тогда, когда каждая КС-грамматика, порождающая его, обладает этим свойством.

Наконец, мы рассмотрим специальные типы КС-грамматик, такие, как последовательные и линейные грамматики.

Формальное определение КС-грамматики допускает структуры, которые в некотором смысле являются "расточительными". Например, словарь может включать нетерминалы, которые не могут использоваться в выводе хоть какой-нибудь терминальной цепочки; или в множестве правил не запрещено иметь такое правило, как $A \to A$. Мы докажем несколько теорем, которые показывают, что каждый КС-язык может порождаться КС-грамматикой специального вида. Более того, будут даны алгоритмы, которые для любой КС-грамматики находят эквивалентную КС-грамматику в одной из заданных форм.

Прежде всего мы докажем результат, который важен сам по себе. Будем предполагать, что КС-грамматики, рассматриваемые в этой главе, не содержат ε-правил.

Теорема 4.1. *Существует алгоритм для определения, является ли язык, порождаемый данной КС-грамматикой, пустым.*

Доказательство. Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. Предположим, что $S \overset{*}{\Rightarrow} x$ для некоторой терминальной цепочки x. Рассмотрим дерево вывода, представляющее этот вывод. Предположим, что в этом дереве есть путь с узлами n_1 и n_2, имеющими одну и ту же метку A. Пусть узел n_1 расположен ближе к корню S, чем узел n_2 (рис. 4.1, *а*).

Поддерево с корнем n_1 представляет вывод $A \overset{*}{\Rightarrow} x_1$ цепочки x_1. Аналогично поддерево с корнем n_2 представляет вывод $A \overset{*}{\Rightarrow} x_2$ цепочки x_2. Заметим, что x_2 является подцепочкой цепочки x_1, которая, впрочем, может совпадать с x_1. Кроме того, цепочка $x = x_3x_1x_4$, где $x_3, x_4 \in \Sigma^*$, причем одна из них или обе могут быть пустыми цепочками. Если в дереве с корнем S мы заменим поддерево с корнем n_1 поддеревом с корнем n_2, то получим дерево (см. рис. 4.1, *б*), представляющее

вывод $S \overset{*}{\Rightarrow} x_3x_2x_4$. Так мы исключили, по крайней мере, один узел (n_1) из исходного дерева вывода.

Если в полученном дереве имеется путь с двумя узлами, помеченными одним и тем же нетерминалом, процесс может быть повторен с деревом вывода $S \overset{*}{\Rightarrow} x_3x_2x_4$. Фактически процесс может повторяться до тех пор, пока в очередном дереве имеется путь, в котором находятся два узла, помеченных одинаково.

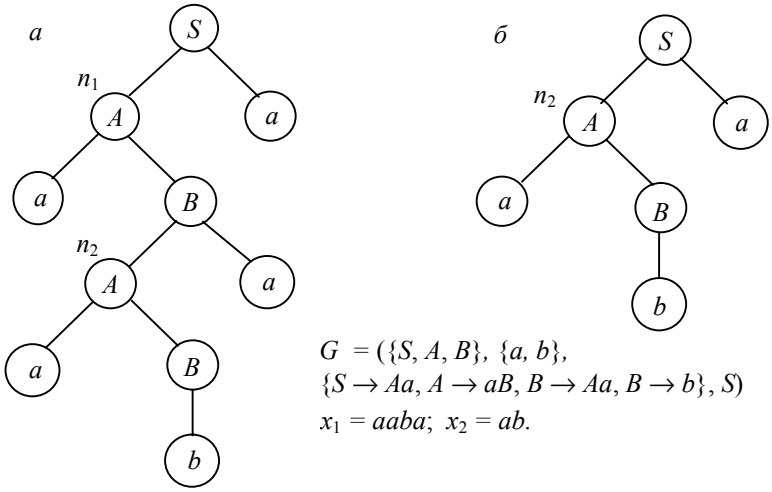

Рис. 4.1.

Поскольку каждая итерация исключает один узел или более, а дерево конечно, то процесс в конце концов закончится. Если в грамматике G имеется m нетерминалов, то в полученном дереве все ветви будут иметь длину (подразумевается, что длина ветви измеряется числом дуг, ее составляющих) не больше m, ибо в противном случае на длинной ветви неминуемо встретились бы два узла с идентичными метками.

Итак, если грамматика G порождает какую-нибудь цепочку вообще, то существует вывод (другой) цепочки, дерево которого не содержит ни одной ветви, длиннее m.

Алгоритм, определяющий, является ли язык $L(G)$ пустым, можно организовать следующим образом. Сначала надо построить коллекцию деревьев, представляющих выводы в грамматике G:

Шаг 1. Начать коллекцию с единственного дерева, представленного только корнем — узлом с меткой S.

Шаг 2. Добавить к коллекции любое дерево, которое может быть получено из дерева, уже имеющегося в коллекции, посредством применения единственного правила, если образующееся дерево не имеет ни одной ветви, длиннее m, и если такого еще нет в коллекции. Поскольку число таких деревьев конечно, то процесс в конце концов закончится.

Шаг 3. Теперь язык $L(G)$ непуст, если в построенной коллекции есть хотя бы одно дерево, представляющее вывод терминальной цепочки. Иначе язык $L(G)$ пуст.

Требуемый алгоритм построен.

Существование алгоритма для определения, порождает ли данная КС-грамматика пустой язык, является важным фактом. Мы будем использовать его при упрощении КС-грамматик.

Как увидим в дальнейшем, никакого такого алгоритма для более сложных грамматик, например для контекстно-зависимых, не существует.

Теорема 4.2. *Для любой контекстно-свободной грамматики* $G = (V_N, V_T, P, S)$, *порождающей непустой язык, можно найти эквивалентную контекстно-свободную грамматику* G_1, *в которой для любого нетерминала A существует терминальная цепочка x, такая, что* $A \overset{*}{\underset{G_1}{\Rightarrow}} x$.

Доказательство. Для каждого нетерминала $A \in V_N$ рассмотрим грамматику $G_A = (V_N, V_T, P, A)$. Если язык $L(G_A)$ пуст, то мы удалим A из алфавита V_N, а также все правила, использующие A в правой или левой части правила. После удаления из G всех таких нетерминалов мы получим новую грамматику: $G_1 = (V_N^1, V_T, P_1, S)$, где V_N^1 и P_1 — оставшиеся нетерминалы и правила. Ясно, что $L(G_1) \subseteq L(G)$, поскольку вывод в G_1 есть также вывод в G.

Предположим, что существует терминальная цепочка $x \in L(G)$, но $x \notin L(G_1)$. Тогда вывод $S \overset{*}{\underset{G}{\Rightarrow}} x$ должен включать сентенциальную форму вида $\alpha_1 A \alpha_2$, где $A \in V_N \setminus V_N^1$, т.е. $S \overset{*}{\underset{G}{\Rightarrow}} \alpha_1 A \alpha_2 \overset{*}{\underset{G}{\Rightarrow}} x$. Однако тогда должна существовать некоторая терминальная цепочка x_1, такая, что $A \overset{*}{\underset{G}{\Rightarrow}} x_1$, — факт, противоречащий предположению о том, что $A \in V_N \setminus V_N^1$. Что и требовалось доказать.

***Определение* 4.1.** Нетерминалы из V_N^1 принято называть *продуктивными.*

В дополнение к исключению нетерминалов, из которых невозможно вывести ни одной терминальной цепочки, мы можем также исключать нетерминалы, которые не участвуют ни в каком выводе.

Теорема 4.3. *Для любой данной контекстно-свободной грамматики, порождающей непустой язык L, можно найти контекстно-свободную грамматику, порождающую язык L, такую, что для каждого ее нетерминала A существует вывод вида* $S \overset{*}{\Rightarrow} x_1 A x_3 \overset{*}{\Rightarrow} x_1 x_2 x_3$, *где* $x_1, x_2, x_3 \in V_T^*$.

Доказательство. Пусть $G_1 = (V_N, V_T, P, S)$ — произвольная cfg, удовлетворяющая условиям теоремы 4.2. Если $S \overset{*}{\underset{G_1}{\Rightarrow}} \alpha_1 A \alpha_2$, где $\alpha_1, \alpha_2 \in V^*$, то существует вывод $S \overset{*}{\underset{G_1}{\Rightarrow}} \alpha_1 A \alpha_2 \overset{*}{\underset{G_1}{\Rightarrow}} x_1 A x_3 \overset{*}{\underset{G_1}{\Rightarrow}} x_1 x_2 x_3$, поскольку терминальные цепочки могут быть выведены из A и из всех нетерминалов, появляющихся в α_1 и α_2. Мы можем эффективно построить множество V_N' всех нетерминалов A, таких, что будет существовать вывод $S \overset{*}{\underset{G_1}{\Rightarrow}} \alpha_1 A \alpha_2$, следующим образом.

Для начала поместим S в искомое множество. Затем последовательно будем добавлять к этому множеству любой нетерминал, который появляется в правой части любого правила из P, определяющего нетерминал, уже имеющийся в этом множестве. Процесс завершается, когда никакие новые элементы не могут быть добавлены к упомянутому множеству.

Положим $G_2 = (V_N', V_T, P', S)$, где P'— множество правил, оставшихся после исключения всех правил из P, которые используют символы из $V_N \setminus V_N'$ слева или справа. G_2 — требуемая грамматика.

Покажем, что $L(G_1) = L(G_2)$ и G_2 удовлетворяет условию теоремы.

I. $L(G_1) \subseteq L(G_2)$. Пусть $x \in L(G_1)$, т.е. $S \overset{*}{\underset{G_1}{\Rightarrow}} x$. Очевидно, что все нетерминалы, встречающиеся в сентенциальных формах этого вывода достижимы, т.е. принадлежат алфавиту V_N', и соответственно в нем участвуют только правила из P'. Следовательно, $S \overset{*}{\underset{G_2}{\Rightarrow}} x$ и $x \in L(G_2)$.

II. $L(G_2) \subseteq L(G_1)$. Это очевидно, так как $P' \subseteq P$.

Из I и II следует, что $L(G_1) = L(G_2)$.

Если $A \in V_N'$, то существует вывод вида $S \overset{*}{\underset{G_2}{\Rightarrow}} \alpha_1 A \alpha_2$, и поскольку все нетерминалы продуктивны, то $S \overset{*}{\underset{G_2}{\Rightarrow}} \alpha_1 A \alpha_2 \overset{*}{\underset{G_2}{\Rightarrow}} x_1 A x_3 \overset{*}{\underset{G_2}{\Rightarrow}} x_1 x_2 x_3$, где $x_1, x_2, x_3 \in V_T^*$.

Что и требовалось доказать.

***Определение* 4.2.** Контекстно-свободные грамматики, удовлетворяющие условию теоремы 4.3, принято называть *приведенными*.

***Определение* 4.3.** Вывод в контекстно-свободной грамматике назовем *левосторонним*, если на каждом его шаге производится замена крайнего левого вхождения нетерминала. Более формально: пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. Вывод в грамматике G вида $S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$ — левосторонний, если для $i = 1, 2, \ldots, n-1$ имеет место $\alpha_i = x_i A_i \beta_i$, $x_i \in V_T^*$, $A_i \in V_N$, $\beta_i \in V^*$, а $A_i \to \gamma_i \in P$. Наконец, $\alpha_{i+1} = x_i \gamma_i \beta_i$, т.е. α_{i+1} выведено из α_i заменой A_i на γ_i.

Для обозначения одного шага или нескольких шагов левостороннего вывода будем использовать значок $\underset{lm}{\Rightarrow}$ или $\overset{*}{\underset{lm}{\Rightarrow}}$ соответственно.

Лемма 4.1. *Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. Если $S \overset{*}{\underset{G}{\Rightarrow}} x$, где $x \in V_T$, то существует и левосторонний вывод $S \overset{*}{\underset{lm}{\Rightarrow}} x$.*

Доказательство. Индукцией по длине вывода l докажем более общее утверждение: если для любого нетерминала $A \in V_N$ существует вывод $A \overset{*}{\underset{G}{\Rightarrow}} x$, то существует и левосторонний вывод $A \overset{*}{\underset{lm}{\Rightarrow}} x$. Утверждение леммы будет следовать как частный случай при $S = A$.

База. Пусть $l = 1$. Для одношагового вывода утверждение выполняется тривиальным образом.

Индукционная гипотеза. Предположим, что утверждение справедливо для любых выводов длиной $l \le n$ ($n \ge 1$).

Индукционный переход. Докажем, что оно справедливо и для $l = n + 1$. Пусть $A \Rightarrow \alpha \overset{*}{\Rightarrow} x$ — вывод длиной $n + 1$ и пусть $\alpha = B_1 B_2 \ldots B_m$, где $B_i \in V^*$, $1 \le i \le m$.

Очевидно, что вывод имеет вид $A \Rightarrow B_1 B_2 \ldots B_m \overset{*}{\Rightarrow} x_1 x_2 \ldots x_m$, причем $B_i \overset{l_i}{\Rightarrow} x_i$, $l_i \le n$, $1 \le i \le m$. Заметим, что некоторые B_i могут быть терминалами, и в этом случае $B_i = x_i$ и вывод не занимает никаких шагов. Если же $B_i \in V_N$, то согласно индукционному предположению $B_i \underset{lm}{\overset{*}{\Rightarrow}} x_i$. Таким образом, мы можем выстроить левосторонний вывод $A \Rightarrow B_1 B_2 \ldots B_m \underset{lm}{\overset{*}{\Rightarrow}} x_1 B_2 \ldots B_m \underset{lm}{\overset{*}{\Rightarrow}} x_1 x_2 \ldots \underset{lm}{\overset{*}{\Rightarrow}} x_1 x_2 \ldots x_m = x$, воспользовавшись частичными левосторонними выводами для тех B_i, которые являются нетерминалами, применяя их в последовательности слева направо. Что и требовалось доказать.

Теорема 4.4. *Любой контекстно-свободный язык может быть порожден контекстно-свободной грамматикой, не содержащей цепных правил, т.е. правил вида $A \to B$, где A и B — нетерминалы.*

Доказательство. Пусть $G = (V_N, V_T, P, S)$ — cfg и $L = L(G)$. Мы построим новое множество правил P_1, прежде всего включив в него все нецепные правила из P. Затем мы добавим в P_1 правила вида $A \to \alpha$ при условии, что существует вывод вида $A \underset{G}{\overset{*}{\Rightarrow}} B$, где A и B — нетерминалы, а $B \to \alpha$ — нецепное правило из P.

Заметим, что мы легко можем проверить, существует ли вывод $A \underset{G}{\overset{*}{\Rightarrow}} B$, поскольку, если $A \underset{G}{\Rightarrow} B_1 \underset{G}{\Rightarrow} B_2 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} B_m \underset{G}{\Rightarrow} B$ и некоторый нетерминал появляется дважды в этом выводе, то мы можем найти более короткую последовательность цепных правил, которая дает результат $A \underset{G}{\overset{*}{\Rightarrow}} B$. Таким образом, достаточно рассматривать только те цепные выводы, длина которых меньше, чем число нетерминалов в V_N.

Мы теперь имеем модифицированную грамматику $G_1 = (V_N, V_T, P_1, S)$.

I. Покажем, что $L(G_1) \subseteq L(G)$. Действительно, если $A \to \alpha \in P_1$, то $A \underset{G}{\overset{*}{\Rightarrow}} \alpha$. Следовательно, если терминальная цепочка x выводится в G_1, то она выводима и в G.

II. Покажем теперь, что $L(G) \subseteq L(G_1)$.

Пусть $x \in L(G)$. Рассмотрим левосторонний вывод $S = \alpha_0 \underset{G}{\Rightarrow} \alpha_1 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} \alpha_n = x$. Если $\alpha_i \underset{G}{\Rightarrow} \alpha_{i+1}$ для $0 \le i < n$ посредством нецепного правила, то $\alpha_i \underset{G_1}{\Rightarrow} \alpha_{i+1}$. Предположим, что $\alpha_i \underset{G}{\Rightarrow} \alpha_{i+1}$ посредством цепного правила, но что $\alpha_{i-1} \underset{G}{\Rightarrow} \alpha_i$ с помощью нецепного правила при условии, конечно, что $i \ne 0$.

Предположим также, что $\alpha_{i+1} \underset{G}{\Rightarrow} \alpha_{i+2} \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} \alpha_j$ все посредством цепных правил, а $\alpha_j \underset{G}{\Rightarrow} \alpha_{j+1}$ при помощи нецепного правила. Тогда все α_{i+1}, α_{i+2}, ..., α_j

одинаковой длины, и поскольку вывод — левосторонний, то нетерминал, заменяемый в каждой из них, должен быть в одной и той же позиции. Но тогда $\alpha_i \underset{G_1}{\Rightarrow} \alpha_{j+1}$ посредством одного из правил из $P_1 \setminus P$. Следовательно, $x \in L(G_1)$.

Из утверждений I и II следует $L(G_1) = L(G)$. Что и требовалось доказать.

§ 4.2. Нормальная форма Хомского

Докажем первую из двух теорем о нормальных формах КС-грамматик. Каждая из них утверждает, что все КС-грамматики эквивалентны грамматикам с ограничениями на вид правил.

Теорема 4.5 — нормальная форма Хомского. *Любой КС-язык может быть порожден грамматикой, в которой все правила имеют вид* $A \to BC$ *или* $A \to a$ (A, B, C *— нетерминалы*, a *— терминал*).

Доказательство. Пусть G — КС-грамматика и $L = L(G)$. В соответствии с теоремой 4.4 мы можем найти эквивалентную cfg $G_1 = (V_N, V_T, P, S)$, такую, что множество ее правил P не содержит ни одного цепного правила. Таким образом, если правая часть правила состоит из одного символа, то этот символ — терминал, и это правило уже находится в приемлемой форме.

Теперь рассмотрим правило в P вида $A \to B_1B_2 \ldots B_m$, где $B_i \in V$, $i = 1, 2, \ldots, m$, $m \geq 2$. Если $B_i \in V_T$, заменим его на новый нетерминал C_i, $C_i \notin V_N$, и создадим новое правило для него вида $C_i \to B_i$, которое имеет допустимую форму, поскольку B_i — терминал. Правило $A \to B_1B_2 \ldots B_m$ заменяется правилом $A \to C_1C_2 \ldots C_m$, где $C_i = B_i$, если $B_i \in V_N$.

Пусть пополненное множество нетерминалов — V_N^2, а пополненное множество правил — P_2.

Рассмотрим грамматику $G_2 = (V_N^2, V_T, P_2, S)$. Пока не все ее правила удовлетворяют нормальной форме Хомского (НФХ). Покажем, что она эквивалентна грамматике G_1.

I. Докажем, что $L(G_1) \subseteq L(G_2)$. Пусть $S \overset{*}{\underset{G_1}{\Rightarrow}} x$. Один шаг этого вывода в грамматике G_1, на котором используется правило $A \to B_1B_2 \ldots B_m \in P$, равносилен в грамматике G_2 применению нового правила: $A \to C_1C_2 \ldots C_m \in P_2$ и нескольких правил вида $C_i \to B_i \in P_2$, о которых шла речь. Поэтому имеем $S \overset{*}{\underset{G_2}{\Rightarrow}} x$.

II. Докажем, что $L(G_2) \subseteq L(G_1)$. Индукцией по длине вывода l покажем, что если для любого $A \in V_N$ существует вывод $A \overset{l}{\underset{G_2}{\Rightarrow}} x$, где $x \in V_T^*$, то $A \overset{*}{\underset{G_1}{\Rightarrow}} x$.

База. Пусть $l = 1$. Если $A \underset{G_2}{\Rightarrow} x$, $A \in V_N$, $x \in V_T^*$, то согласно построению грамматики G_2 использованное правило $A \to x \in P_2$ имеется также и во множестве правил P. Действительно, $|x|$ не может быть больше единицы, так как такое пра-

правило не могло бы быть в множестве правил P_2. Следовательно, x — просто терминал, и $A \to x \in P$, а тогда $A \underset{G_1}{\Rightarrow} x$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $1 \le l \le n$ $(n \ge 1)$.

Индукционный переход. Пусть $A \overset{l}{\underset{G_2}{\Rightarrow}} x$, где $l = n + 1$. Этот вывод имеет вид: $A \underset{G_2}{\Rightarrow} C_1 C_2 \ldots C_m \overset{n}{\underset{G_2}{\Rightarrow}} x$, где $m \ge 2$.

Очевидно, что $x = x_1 x_2 \ldots x_m$, причем $C_i \overset{l_i}{\underset{G_2}{\Rightarrow}} x_i$, $l_i \le n$, $i = 1, 2, \ldots, m$. Если $C_i \in V_N^2 \setminus V_N$, то существует только одно правило из множества правил P_2, которое определяет этот нетерминал: $C_i \to a_i$ для некоторого $a_i \in V_T$. В этом случае $a_i = x_i$. По построению правило $A \to C_1 C_2 \ldots C_m \in P_2$, используемое на первом шаге вывода, обязано своим происхождением правилу $A \to B_1 B_2 \ldots B_m \in P$, где $B_i = C_i$, если $C_i \in V_N$, и $B_i = a_i$, если $C_i \in V_N^2 \setminus V_N$. Для $C_i \in V_N$ мы имеем выводы $C_i \overset{l_i}{\underset{G_2}{\Rightarrow}} x_i$, $l_i \le n$, и по индукционному предположению существуют выводы $B_i \overset{*}{\underset{G_1}{\Rightarrow}} x_i$. Следовательно, $A \overset{*}{\underset{G_1}{\Rightarrow}} x$. При $A = S$ имеем как частный случай $x \in L(G_1)$.

Итак, мы доказали промежуточный результат: любой контекстно-свободный язык может быть порожден контекстно-свободной грамматикой, каждое правило которой имеет форму $A \to a$ или $A \to B_1 B_2 \ldots B_m$, где $m \ge 2$; $A, B_1, B_2, \ldots, B_m$ — нетерминалы; a — терминал.

Очевидно, что все правила при $m \le 2$ имеют такой вид, какого требует нормальная форма Хомского. Остается преобразовать правила для $m \ge 3$ к надлежащему виду. Если $G_2 = (V_N^2, V_T, P_2, S)$ — такая cfg, модифицируем ее, добавляя некоторые дополнительные нетерминалы и заменяя некоторые ее правила. Именно: для каждого правила вида $A \to B_1 B_2 \ldots B_m \in P_2$, где $m \ge 3$, мы создаем новые нетерминалы $D_1, D_2, \ldots, D_{m-2} \notin V_N^2$ и заменяем правило $A \to B_1 B_2 \ldots B_m \in P_2$ множеством правил $\{A \to B_1 D_1,\ D_1 \to B_2 D_2, \ldots,\ D_{m-3} \to B_{m-2} D_{m-2},\ D_{m-2} \to B_{m-1} B_m\}$. Пусть V_N^3 — новый нетерминальный словарь, а P_3 — новое множество правил.

Рассмотрим контекстно-свободную грамматику $G_3 = (V_N^3, V_T, P_3, S)$. Докажем, что она эквивалентна грамматике G_2.

III. Докажем, что $L(G_2) \subseteq L(G_3)$. Пусть $S \overset{*}{\underset{G_1}{\Rightarrow}} x$. Один шаг этого вывода в грамматике G_2, на котором используются правила вида $A \to a$ или $A \to B_1 B_2$, является и шагом вывода в грамматике G_3, так как по построению эти правила также входят в грамматику G_3.

Шаг вывода в грамматике G_2, на котором используется правило $A \to B_1 B_2 \ldots B_m \in P_2$, $m \ge 3$, равносилен последовательному применению правил $A \to B_1 D_1$, $D_1 \to B_2 D_2, \ldots, D_{m-3} \to B_{m-2} D_{m-2}, D_{m-2} \to B_{m-1} B_m \in P_3$. Поэтому имеем $S \overset{*}{\underset{G_3}{\Rightarrow}} x$.

IV. Докажем, что $L(G_3) \subseteq L(G_2)$. Индукцией по длине вывода l покажем, что если для любого $A \in V_N$ существует вывод $A \overset{l}{\underset{G_3}{\Rightarrow}} x$, $x \in V_T^*$, то $A \overset{*}{\underset{G_2}{\Rightarrow}} x$.

База. Пусть $l = 1$. Если $A \underset{G_3}{\Rightarrow} x$, $A \in V_N$, $x \in V_T^*$, то согласно построению G_3 использованное правило $A \to x \in P_3$ содержится также и во множестве правил P_2, так как в этом случае $x \in V_T$, а тогда $A \overset{*}{\underset{G_2}{\Rightarrow}} x$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $1 \le l \le n$ $(n \ge 1)$.

Индукционный переход. Пусть $A \overset{l}{\underset{G_3}{\Rightarrow}} x$, где $l = n + 1$. Этот вывод имеет следующий вид: $A \underset{G_3}{\Rightarrow} B_1D_1 \underset{G_3}{\Rightarrow} B_1B_2D_2 \underset{G_3}{\Rightarrow} \ldots \underset{G_3}{\Rightarrow} B_1B_2 \ldots B_{m-2}D_{m-2} \underset{G_3}{\Rightarrow} B_1B_2 \ldots B_m \overset{*}{\underset{G_3}{\Rightarrow}} x$. Очевидно, что $x = x_1x_2 \ldots x_m$, где $B_i \overset{l_i}{\underset{G_3}{\Rightarrow}} x_i$, $l_i \le n$, $i = 1, 2, \ldots, m$. По индукционной гипотезе $B_i \overset{*}{\underset{G_2}{\Rightarrow}} x_i$. Следовательно, $A \overset{*}{\underset{G_2}{\Rightarrow}} x$. В частности, при $A = S$ получаем $S \overset{*}{\underset{G_2}{\Rightarrow}} x$. Утверждение IV доказано, а вместе с ним доказано равенство $L(G_2) = L(G_3)$, и сама теорема.

Пример 4.1. Рассмотрим грамматику $G = (\{S, A, B\}, \{a, b\}, P, S), \{a, b\}, S)$, в которой $P = \{S \to bA,\ S \to aB,\ A \to a,\ A \to aS,\ A \to bAA,\ B \to b,\ B \to bS,\ B \to aBB\}$.

Построим эквивалентную грамматику в нормальной форме Хомского. Во-первых, два правила, а именно: $A \to a$ и $B \to b$, уже имеют требуемый вид. Нет никаких цепных правил, так что мы можем начать с замены терминалов в правых частях остальных правил на новые нетерминалы и построения правил для них. Правило $S \to bA$ заменяется двумя правилами $S \to C_1A$, $C_1 \to b$. Аналогично правило $S \to aB$ заменяется правилами $S \to C_2B$, $C_2 \to a$. Вместо $A \to aS$ вводятся правила $A \to C_3S$, $C_3 \to a$. Правило $A \to bAA$ заменяется тремя новыми $A \to C_4D_1$, $C_4 \to b$, $D_1 \to AA$. Правило $B \to bS$ заменяется правилами $B \to C_5S$, $C_5 \to b$. Правило $B \to aBB$ заменяется правилами $C_6 \to a$, $B \to C_6D_2$, $D_2 \to BB$.

Итак, мы получили эквивалентную грамматику в НФХ:

$G_1 = (\{S, A, B, C_1, C_2, C_3, C_4, C_5, C_6, D_1, D_2\}, \{a, b\}, P_1, S), \{a, b\}, S)$, где

$P_1 = \{S \to C_1A, S \to C_2B, A \to C_3S\ A \to C_4D_1, A \to a, B \to C_5S, B \to C_6D_2, B \to b,$
$C_1 \to b, C_2 \to a, C_3 \to a, C_4 \to b, C_5 \to b, C_6 \to a, D_1 \to AA, D_2 \to BB\}$.

§ 4.3. Нормальная форма Грейбах

Определение **4.4.** Говорят, что контекстно-свободная грамматика $G = (V_N, V_T, P, S)$ представлена в *нормальной форме Грейбах*, если каждое ее правило имеет вид $A \to a\alpha$, где $a \in V_T$, $a \in V_N^*$.

Для доказательства того, что всякая контекстно-свободная грамматика может быть приведена к нормальной форме Грейбах, нам потребуется обосновать эквивалентность используемых при этом преобразований.

Лемма 4.2. *Пусть* $G = (V_N, V_T, P, S)$ *— контекстно-свободная грамматика*, $A \to \alpha_1 B \alpha_2 \in P$ *и* $\{B \to \beta_i \mid B \in V_N, \beta_i \in V^+, i = 1, 2, \ldots, m\}$ *— множество всех B-порождений, т.е. правил с нетерминалом B в левой части. Пусть грамматика* $G_1 = (V_N, V_T, P_1, S)$ *получается из грамматики G отбрасыванием правила* $A \to \alpha_1 B \alpha_2$ *и добавлением правил вида* $A \to \alpha_1 \beta_i \alpha_2$, $i = 1, 2, \ldots, m$. *Тогда* $L(G) = L(G_1)$.

Доказательство.

I. Очевидно, что $L(G_1) \subseteq L(G)$. Пусть $S \overset{*}{\underset{G_1}{\Rightarrow}} x$, $x \in V_T^*$. Использование в этом выводе правила $A \to \alpha_1 \beta_i \alpha_2 \in P_1 \setminus P$ равносильно двум шагам вывода в грамматике G: $A \underset{G}{\Rightarrow} \alpha_1 B \alpha_2 \underset{G}{\Rightarrow} \alpha_1 \beta_i \alpha_2$. Шаги вывода в грамматике G_1, на которых используются другие правила из множества правил P, являются шагами вывода в грамматике G. Поэтому $S \overset{*}{\underset{G}{\Rightarrow}} x$.

II. Очевидно, что $L(G) \subseteq L(G_1)$. Пусть $S \overset{*}{\underset{G}{\Rightarrow}} x$. Если в этом выводе используется правило $A \to \alpha_1 B \alpha_2 \in P \setminus P_1$, то рано или поздно для замены B будет использовано правило вида $B \to \beta_i \in P$. Эти два шага вывода в грамматике G равносильны одному шагу вывода в грамматике G_1: $A \underset{G_1}{\Rightarrow} \alpha_1 \beta_i \alpha_2$. Шаги вывода в грамматике G, на которых используются другие правила из множества P, являются шагами вывода в грамматике G_1. Поэтому $S \overset{*}{\underset{G_1}{\Rightarrow}} x$. Что и требовалось доказать.

Лемма 4.3 — об устранении левой рекурсии. *Пусть* $G = (V_N, V_T, P, S)$ *— контекстно-свободная грамматика*, $\{A \to A\alpha_i \mid A \in V_N, \alpha_i \in V^*, i = 1,2, \ldots, m\}$ *— множество всех леворекурсивных A-порождений*, $\{A \to \beta_j \mid j = 1,2, \ldots, n\}$ *— множество всех прочих A-порождений*.

Пусть $G_1 = (V_N \cup \{Z\}, V_T, P_1, S)$ *— контекстно-свободная грамматика, образованная добавлением нетерминала Z к* V_N *и заменой всех A-порождений правилами*:

1) $A \to \beta_j$, $A \to \beta_j Z$, $j = 1, 2, \ldots, n$;
2) $Z \to \alpha_i$, $Z \to \alpha_i Z$, $i = 1, 2, \ldots, m$.

Тогда $L(G_1) = L(G)$.

Доказательство. Прежде всего заметим, что посредством левосторонних выводов при использовании одних лишь A-порождений порождаются регулярные множества вида $\{\beta_1, \beta_2, \ldots, \beta_n\}\{\alpha_1, \alpha_2, \ldots, \alpha_m\}^*$, и это является в точности множеством, порождаемым правилами грамматики G_1, имеющими A или Z в левых частях.

I. Докажем, что $L(G) \subseteq L(G_1)$. Пусть $x \in L(G)$. Левосторонний вывод $S \overset{*}{\underset{G}{\Rightarrow}} x$ мы можем перестроить в вывод $S \overset{*}{\underset{G_1}{\Rightarrow}} x$ следующим образом: каждый раз, когда в левостороннем выводе встречается последовательность шагов:

$$tA\gamma \underset{G}{\Rightarrow} tA\alpha_{i_1}\gamma \underset{G}{\Rightarrow} tA\alpha_{i_2}\alpha_{i_1}\gamma \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} tA\alpha_{i_p} \ldots \alpha_{i_2}\alpha_{i_1}\gamma \underset{G}{\Rightarrow} t\beta_j\alpha_{i_p} \ldots \alpha_{i_2}\alpha_{i_1}\gamma$$

($t \in V_T^*$, $\gamma \in V^*$), заменим их последовательностью

$$tA\gamma \underset{G_1}{\Rightarrow} t\beta_j Z\gamma \underset{G_1}{\Rightarrow} t\beta_j\alpha_{i_p}Z\gamma \underset{G_1}{\Rightarrow} \ldots \underset{G_1}{\Rightarrow} t\beta_j\alpha_{i_p} \ldots \alpha_{i_2}Z\gamma \underset{G_1}{\Rightarrow} t\beta_j\alpha_{i_p} \ldots \alpha_{i_2}\alpha_{i_1}\gamma.$$

Полученный таким образом вывод является выводом цепочки x в грамматике G_1, хотя и не левосторонним. Следовательно, $x \in L(G_1)$.

II. Докажем, что $L(G_1) \subseteq L(G)$. Пусть $x \in L(G_1)$. Рассмотрим левосторонний вывод $S \overset{*}{\underset{G_1}{\Rightarrow}} x$, и перестроим его в вывод в грамматике G следующим образом. Всякий раз, как Z появляется в сентенциальной форме, мы приостанавливаем левосторонний порядок вывода и вместо того, чтобы производить замены в цепочке β, предшествующей Z, займемся замещением Z с помощью правил вида $Z \to \alpha Z$. Далее, вместо того, чтобы производить подстановки в цепочке α, продолжим использовать соответствующие правила для Z, пока, наконец, Z не будет замещено цепочкой, его не содержащей. После этого можно было бы заняться выводами терминальных цепочек из β и α. Результат этого, уже не левостороннего, вывода будет тем же самым, что и при исходном левостороннем выводе в грамматике G_1.

В общем случае вся последовательность шагов этого перестроенного участка вывода, в которых участвует Z, имеет вид

$$tA\gamma \underset{G_1}{\Rightarrow} t\beta_j Z\gamma \underset{G_1}{\Rightarrow} t\beta_j\alpha_{i_p} Z\gamma \underset{G_1}{\Rightarrow} \ldots \underset{G_1}{\Rightarrow} t\beta_j\alpha_{i_p} \ldots \alpha_{i_2} Z\gamma \underset{G_1}{\Rightarrow} t\beta_j\alpha_{i_p} \ldots \alpha_{i_2}\alpha_{i_1}\gamma.$$

Очевидно, что такой же результат может быть получен в грамматике G:

$$tA\gamma \underset{G}{\Rightarrow} tA\alpha_{i_1}\gamma \underset{G}{\Rightarrow} tA\alpha_{i_2}\alpha_{i_1}\gamma \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} tA\alpha_{i_p} \ldots \alpha_{i_2}\alpha_{i_1}\gamma \underset{G}{\Rightarrow} t\beta_j\alpha_{i_p} \ldots \alpha_{i_2}\alpha_{i_1}\gamma.$$

Таким образом, $L(G_1) = L(G)$. Что и требовалось доказать.

Теорема 4.6 — нормальная форма Грейбах. *Каждый контекстно-свободный язык может быть порожден контекстно-свободной грамматикой в нормальной форме Грейбах.*

Доказательство. Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика в нормальной форме Хомского, порождающая контекстно-свободный язык L. Пусть $V_N = \{A_1, A_2, \ldots, A_m\}$.

Первый шаг построения состоит в том, чтобы в правилах вида $A_i \to A_j\gamma$, где γ — цепочка нетерминалов новой грамматики, всегда было $j > i$. Этот шаг выполняется последовательно для $i = 1, 2, \ldots, m$ следующим образом.

При $i = 1$ правило для A_1 может иметь вид $A_1 \to a$, $a \in V_T$, и тогда оно не нуждается в преобразованиях, либо оно имеет вид $A_1 \to A_jA_k$, A_j, $A_k \in V_N$. Если $j > 1$, то правило уже имеет требуемый вид. В противном случае оно леворекурсивно, и в соответствии с леммой 4.3 может быть заменено правилами вида $A_1 \to \beta$, $A_1 \to \beta Z_1$, $Z_1 \to A_k$, $Z_1 \to A_kZ_1$, $\beta = a$, $a \in V_T$, или $\beta = BC$, причем $B \neq A_1$.

Предположим, что для $i = 1, 2, \ldots, k$ правила вида $A_i \to A_j\gamma$ были преобразованы так, что $j > i$.

Покажем, как добиться выполнения этого условия для A_{k+1}-порождений. Если $A_{k+1} \to A_j\gamma$ есть правило, в котором $j < k + 1$, то мы образуем новые правила, подставляя вместо A_j правую часть каждого A_j-порождения согласно лемме 4.2. В результате, если в позиции A_j окажется нетерминал, то его номер будет больше j. Повторив этот процесс самое большее $k-1$ раз, получим порождения вида $A_{k+1} \to A_p\gamma$, $p \geq k + 1$. Порождения с $p = k + 1$ затем преобразуются согласно лемме 4.3 введением новой переменной Z_{k+1}.

Повторив описанный процесс для каждого нетерминала исходной грамматики, мы получим правила только одного из трех следующих видов:

$A_k \to A_p\gamma$, где $p > k$

$A_k \to a\gamma$, где $a \in V_T$

$Z_k \to X\gamma$, где $X \in V_T \cup V_N$, $\gamma \in (V_N \cup \{ Z_1, Z_2, \ldots, Z_m\})^*$.

Отметим, что крайний левый символ правой части правила для A_m по необходимости является терминалом, так как нетерминала с большим номером не существует. Крайний левый символ в правой части правила для A_{m-1} может быть терминалом либо нетерминалом A_m. В последнем случае мы можем построить новые правила, заменяя A_m правыми частями A_m-порождений согласно лемме 4.2. Эти новые правила будут иметь правые части, начинающиеся с терминального символа.

Подобным же образом преобразуем правила для $A_{m-2}, A_{m-3}, \ldots, A_1$ до тех пор, пока правые части каждого из этих правил не будут начинаться с терминала.

Остается преобразовать правила для новых переменных $Z_1, Z_2, \ldots, Z_m$. Правые части этих правил начинаются с терминального символа либо с нетерминала исходной грамматики. Пусть имеется правило вида $Z_i \to A_k\gamma$. Достаточно еще раз применить к нему преобразования, описанные в лемме 4.2, заменив A_k правыми частями A_k-порождений, чтобы получить требуемую форму правил, поскольку правые части правил для A_k уже начинаются с терминалов. На этом построение грамматики в нормальной форме Грейбах, эквивалентной исходной грамматике G, завершается. Что и требовалось доказать.

Пример 4.2. Преобразуем грамматику $G = (\{A_1, A_2, A_3\}, \{a, b\}, P, A_1)$, где $P = \{A_1 \to A_2A_3,\ A_2 \to A_3A_1,\ A_2 \to b,\ A_3 \to A_1A_2,\ A_3 \to a\}$, к нормальной форме Грейбах.

Шаг 1. Поскольку правые части правил для A_1 и A_2 начинаются с нетерминалов с большими номерами и с терминала, то мы начинаем с правила $A_3 \to A_1A_2$ и подставляем цепочку A_2A_3 вместо A_1. Заметим, что $A_1 \to A_2A_3$ является единственным правилом с A_1 в левой части. В результате получаем следующее множество правил

$$\{A_1 \to A_2A_3\,,\ A_2 \to A_3A_1,\ A_2 \to b,\ A_3 \to A_2A_3A_2\,,\ A_3 \to a\}.$$

Поскольку правая часть правила $A_3 \to A_2A_3A_2$ начинается с нетерминала с меньшим номером, мы подставляем вместо первого вхождения A_2 либо A_3A_1, либо b. Таким образом, правило $A_3 \to A_2A_3A_2$ заменяется на $A_3 \to A_3A_1A_3A_2$ и $A_3 \to bA_3A_2$. Новое множество есть

$$\{A_1 \to A_2A_3,\ A_2 \to A_3A_1,\ A_2 \to b,\ A_3 \to A_3A_1A_3A_2,\ A_3 \to bA_3A_2,\ A_3 \to a\}.$$

Теперь применим лемму 4.3 к правилам $A_3 \to A_3A_1A_3A_2$, $A_3 \to bA_3A_2$ и $A_3 \to a$. Введем символ Z_3 и заменим правило $A_3 \to A_3A_1A_3A_2$ правилами $A_3 \to bA_3A_2Z_3$, $A_3 \to aZ_3$, $Z_3 \to A_1A_3A_2$ и $Z_3 \to A_1A_3A_2Z_3$. Теперь мы имеем множество:

$$\{A_1 \to A_2A_3,\ A_2 \to A_3A_1,\ A_2 \to b,\ A_3 \to bA_3A_2,\ A_3 \to a,$$
$$A_3 \to bA_3A_2Z_3,\ A_3 \to aZ_3, Z_3 \to A_1A_3A_2Z_3,\ Z_3 \to A_1A_3A_2\}.$$

Шаг 2. Все правила с A_3 слева начинаются с терминалов. Они используются для замены A_3 в правиле $A_2 \to A_3A_1$, а затем правила для A_2 используются для того, чтобы заменить A_2 в правиле $A_1 \to A_2A_3$. Получаем:

$A_3 \to bA_3A_2$, $A_2 \to bA_3A_2A_1$, $A_1 \to bA_3A_2A_1A_3$, $Z_3 \to A_1A_3A_2Z_3$,
$A_3 \to a$, $A_2 \to bA_3A_2Z_3A_1$, $A_1 \to bA_3A_2Z_3A_1A_3$, $Z_3 \to A_1A_3A_2$.
$A_3 \to bA_3A_2Z_3$, $A_2 \to aA_1$, $A_1 \to aA_1A_3$,
$A_3 \to aZ_3$; $A_2 \to aZ_3A_1$, $A_1 \to aZ_3A_1A_3$,
$A_2 \to b$; $A_1 \to bA_3$;

Шаг 3. Два правила для Z_3 заменяются на десять новых в результате подстановки в них вместо A_1 правых частей правил для A_1:

$Z_3 \to bA_3A_2A_1A_3A_3A_2Z_3$, $Z_3 \to bA_3A_2A_1A_3A_3A_2$,
$Z_3 \to bA_3A_2Z_3A_1A_3A_3A_2Z_3$, $Z_3 \to bA_3A_2Z_3A_1A_3A_3A_2$,
$Z_3 \to aA_1A_3A_3A_2Z_3$, $Z_3 \to aA_1A_3A_3A_2$,
$Z_3 \to aZ_3A_1A_3A_3A_2Z_3$, $Z_3 \to aZ_3A_1A_3A_3A_2$,
$Z_3 \to bA_3A_3A_2Z_3$, $Z_3 \to bA_3A_3A_2$.

Окончательно, получаем следующее множество правил эквивалентной грамматики в нормальной форме Грейбах:

$A_3 \to bA_3A_2$, $A_2 \to bA_3A_2A_1$, $A_1 \to bA_3A_2A_1A_3$, $Z_3 \to bA_3A_2A_1A_3A_3A_2Z_3$,
$A_3 \to a$, $A_2 \to bA_3A_2Z_3A_1$, $A_1 \to bA_3A_2Z_3A_1A_3$, $Z_3 \to bA_3A_2Z_3A_1A_3A_3A_2Z_3$,
$A_3 \to bA_3A_2Z_3$, $A_2 \to aA_1$, $A_1 \to aA_1A_3$, $Z_3 \to aA_1A_3A_3A_2Z_3$,
$A_3 \to aZ_3$; $A_2 \to aZ_3A_1$, $A_1 \to aZ_3A_1A_3$, $Z_3 \to aZ_3A_1A_3A_3A_2Z_3$,
$A_2 \to b$; $A_1 \to bA_3$; $Z_3 \to bA_3A_3A_2Z_3$,
$Z_3 \to bA_3A_2A_1A_3A_3A_2$,
$Z_3 \to bA_3A_2Z_3A_1A_3A_3A_2$,
$Z_3 \to aA_1A_3A_3A_2$,
$Z_3 \to aZ_3A_1A_3A_3A_2$,
$Z_3 \to bA_3A_3A_2$.

§ 4.4. Разрешимость конечности контекстно-свободных языков

В теореме 4.2 было показано, что из контекстно-свободной грамматики можно исключить те нетерминалы, которые не порождают терминальных цепочек. Фактически можно добиться большего. Мы можем протестировать, является ли язык, порождаемый из данного нетерминала, конечным или бесконечным, и исключить те нетерминалы, не являющиеся начальным нетерминалом грамматики, из которых можно породить только конечное число терминальных цепочек. При доказательстве этого утверждения, мы покажем два результата (теоремы 4.7 и 4.8), очень интересные и сами по себе.

Теорема 4.7 — "*uvwxy*". *Пусть L — контекстно-свободный язык. Существуют постоянные p и q, зависящие только от языка L, такие, что если существует* $z \in L$ *при* $|z| > p$, *то цепочка z представима в виде* $z = uvwxy$, *где* $|vwx| \le q$, *причем v, x одновременно не пусты, так что для любого целого* $i \ge 0$ *цепочка* $uv^i wx^i y \in L$.

Доказательство. Пусть $G = (V_N, V_T, P, S)$ — какая-нибудь контекстно-свободная грамматика в нормальной форме Хомского для языка L. Если $\#V_N = k$, положим $p = 2^{k-1}$ и $q = 2^k$. Докажем теорему для этих значений p и q.

Заметим, что дерево вывода любой терминальной цепочки в грамматике G является бинарным. Поэтому, если в нем нет пути, длиннее j, то выводимая терминальная цепочка не длиннее 2^{j-1}.

Пусть существует $z \in L$, причем $|z| > p = 2^{k-1}$. Тогда самый длинный путь в дереве вывода цепочки z длиннее k, ибо в противном случае $|z| \le 2^{k-1}$, и это противоречило бы предположению, что $|z| > p$.

Рассмотрим самый длинный путь (R) в дереве вывода z (от корня до листа). В нем должны быть два узла: n_1 и n_2, удовлетворяющие следующим условиям:

1) они имеют одинаковые метки, скажем, $A \in V_N$;

2) узел n_1 ближе к корню, чем узел n_2;

3) часть пути R от узла n_1 до листа[5] имеет длину, равную самое большее $k + 1$.

Чтобы убедиться в том, что такие узлы всегда можно найти, пройдем R от листа в сторону корня. Из первых $k + 2$ пройденных узлов только один имеет терминальную метку. Остальные $k + 1$ узлов не могут быть помечены разными нетерминалами.

Рассмотрим поддерево T_1 с корнем n_1. Его результат, являющийся подсловом слова z, обозначим через z_1. В поддереве T_1 не может быть пути, длиннее $k + 1$, так как R является самым длинным путем во всем дереве T. Действительно, пусть $R = Sn_1 + n_1a$. Если допустить, что в T_1 существует другой, более длинный, путь, скажем n_1b, то путь $R' = Sn_1 + n_1b$ окажется длиннее R, так как n_1b длиннее n_1a. Однако это противоречило бы первоначальному предположению, что R является самым длинным путем во всем дереве T. Потому $|z_1| \le 2^k = q$. Эти рассуждения иллюстрирует рис. 4.2.

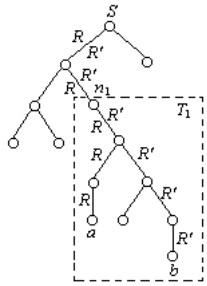

Рис. 4.2.

[5] Очевидно, что самый длинный путь в дереве вывода всегда содержит лист.

Обозначим через T_2 поддерево с корнем n_2, а его результат — через z_2. Ясно, что цепочка z_1 представима в форме $z_1 = z_3 z_2 z_4$, где z_3 и z_4 одновременно не пусты. Действительно, если первое правило, используемое в выводе z_1, имеет вид $A \to BC$, то поддерево T_2 должно быть полностью в пределах либо поддерева с корнем B, либо поддерева с корнем C.

Рис. 4.3 иллюстрирует три случая: (*а*) когда B есть корень поддерева T_2 ($z_3 = \varepsilon$), (*б*) C есть корень поддерева T_2 ($z_4 = \varepsilon$), (*в*) корень поддерева T_2 расположен внутри поддерева B ($z_3 \neq \varepsilon$, $z_4 \neq \varepsilon$).

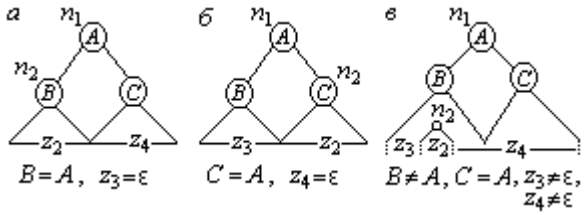

Рис. 4.3.

Теперь мы знаем, что $A \overset{*}{\underset{G}{\Rightarrow}} z_3 A z_4$ и, само собой разумеется, что $A \overset{*}{\underset{G}{\Rightarrow}} z_2$. Поэтому $A \overset{*}{\underset{G}{\Rightarrow}} z_3^i z_2 z_4^i$ для любого $i \geq 0$ и цепочка z представима в виде $z = u z_3 z_2 z_4 y$ для некоторых $u, y \in V_T^*$.

Чтобы закончить доказательство, положим $v = z_3$, $w = z_2$ и $x = z_4$.

Теорема 4.8. *Существует алгоритм для определения, порождает ли данная контекстно-свободная грамматика G конечный или бесконечный язык.*

Доказательство. Пусть p и q — константы, определяемые теоремой 4.7. Если $z \in L(G)$ и $|z| > p$, то $z = uvwxy$ при некоторых $u, v, w, x, y \in V_T^*$, $|v| + |x| > 0$, и для любого $i \geq 0$ цепочка $uv^i wx^i y \in L(G)$. Следовательно, если в языке $L(G)$ существует цепочка длиной больше p, то язык $L(G)$ бесконечен.

Пусть язык $L = L(G)$ бесконечен. Тогда в нем имеются сколь угодно длинные цепочки и, в частности, цепочка длиной больше $p + q$. Эта цепочка может быть представлена как $uvwxy$, где $|vwx| \leq q$, $|v| + |x| > 0$, и цепочка $uv^i wx^i y \in L$ для любого $i \geq 0$. В частности, при $i = 0$ цепочка $uwy \in L$ и $|uwy| < |uvwxy|$.

Убедимся в том, что $|uwy| > p$. Так как $p + q < |uvwxy|$ и $q \geq |vwx|$, то $p < |uy| \leq |uwy|$. Если $|uwy| > p + q$, то эту процедуру можно повторять снова до тех пор, пока мы не найдем цепочку в языке L, длина которой (l) не будет удовлетворять неравенству $p < l \leq p + q$.

Таким образом, язык L бесконечен тогда и только тогда, когда он содержит цепочку длиной l, $p < l \leq p + q$. Поскольку мы можем проверить, имеется ли данная цепочка в данном контекстно-свободном языке L (см. теорему 2.2 о рекурсивности контекстно-зависимых грамматик), то мы просто должны проверять все цепочки в интервале длин между p и $p + q$ на принадлежность языку $L(G)$. Если такая цепочка имеется, то ясно, что язык L бесконечен; если в языке L нет цепочек длиной больше p, то язык L конечен. Что и требовалось доказать.

В теореме 4.2 доказывалось, что из контекстно-свободной грамматики можно исключить все нетерминалы, из которых не выводится ни одной терминальной цепочки. Теперь мы докажем возможность исключения нетерминалов, из которых выводится только конечное число терминальных цепочек.

Теорема 4.9. *Для всякой контекстно-свободной грамматики G_1 можно найти эквивалентную ей контекстно-свободную грамматику G_2, такую, что если A — нетерминал грамматики G_2, не являющийся начальным нетерминалом, то из A выводимо бесконечно много терминальных цепочек.*

Доказательство. Если язык $L(G_1) = \{x_1, x_2, ..., x_n\}$ конечен, то утверждение очевидно. Действительно, положим $G_2 = (\{S\}, V_T, P_2, S)$, где $P_2 = \{S \to x_i \mid i = 1, 2, ..., n\}$. В этой грамматике совсем нет нетерминалов, отличных от S.

Пусть теперь грамматика $G_1 = (V_N, V_T, P_1, S)$ и язык $L(G_1)$ бесконечен. Рассмотрим грамматики $G_A = (V_N, V_T, P_1, A)$ для всех $A \in V_N$. Так как существует алгоритм, позволяющий узнать, бесконечен ли порождаемый грамматикой G_A язык, то весь словарь V_N мы можем разбить на две части: $V_N = \{A_1, A_2, ..., A_k\} \cup \{B_1, B_2, ..., B_m\}$, где A_i $(i = 1, 2, \ldots, k)$ — нетерминалы, порождающие бесконечно много терминальных цепочек, причем начальный нетерминал S среди них, поскольку язык L бесконечен; B_j $(j = 1, 2, \ldots, m)$ — нетерминалы, порождающие конечные множества терминальных цепочек.

Построим грамматику $G_2 = (\{A_1, A_2, ..., A_k\}, V_T, P_2, S)$, где

$P_2 = \{A_i \to u_1 u_2 ... u_r \mid \exists\, A_i \to C_1 C_2 ... C_r \in P_1,$

(1) $u_i = C_i$, если $C_i \in V_T \cup \{A_1, A_2, ..., A_k\}$,

(2) $C_i \overset{*}{\underset{G_1}{\Rightarrow}} u_i$, $u_i \in V_T^*$, если $C_i \in \{B_1, B_2, ..., B_m\}\}$.

Короче говоря, правила P_2 получаются из правил P_1 посредством отбрасывания всех правил с нетерминалами B_j в левых частях, а в оставшихся правилах для нетерминалов A_i все вхождения нетерминалов B_j в правых частях надо заменить какими-нибудь их терминальными порождениями. Поскольку число таких терминальных порождений конечно, то и число получающихся правил в P_2 тоже конечно.

Покажем теперь, что $L(G_1) = L(G_2)$.

I. $L(G_1) \subseteq L(G_2)$. Индукцией по длине вывода l докажем, что если $A_i \overset{l}{\underset{G_1}{\Rightarrow}} w$, то $A_i \overset{*}{\underset{G_2}{\Rightarrow}} w$, $w \in V_T^*$ $(i = 1, 2, ..., k)$.

База. Пусть $l = 1$ и пусть $A_i \underset{G_1}{\Rightarrow} w$. При этом применялось правило $A_i \to w \in P_1$, где $w \in V_T^*$. Но это же правило есть в P_2 по построению. Поэтому $A_i \underset{G_2}{\Rightarrow} w$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех выводов длиной $l \le n$ $(n \ge 1)$.

Индукционный переход. Рассмотрим вывод длиной $l = n + 1$, причем $A_i \underset{G_1}{\Rightarrow} C_1 C_2 ... C_r \overset{n}{\underset{G_1}{\Rightarrow}} w_1 w_2 ... w_r$, где $C_p \overset{l_p}{\underset{G_1}{\Rightarrow}} w_p$, $w_p \in V_T^*$, $l_p \le n$. На первом шаге при-

меняется правило $A_i \to C_1C_2 \dots C_r \in P_1$. Возьмем во множестве правил P_2 соответствующее правило, которое получается из данного заменой в нем всех нетерминалов типа B на соответствующие w_p, т.е. правило $A_i \to u_1u_2 \dots u_r \in P_2$, в котором $u_p = w_p$, если $C_p \in V_T \cup \{B_1, B_2, \dots, B_k\}$, $u_p = C_p$, если $C_p \in \{A_1, A_2, \dots, A_k\}$, $p = 1, 2, \dots, r$.

Таким образом, имеем $A_i \underset{G_2}{\Rightarrow} u_1u_2 \dots u_r$, причем здесь все $u_p = w_p$, кроме тех u_p, которые равны $C_p \in \{A_1, A_2, \dots, A_k\}$. Но для них $u_p = C_p \overset{l_p}{\underset{G_1}{\Rightarrow}} w_p$, $l_p \le n$, и по индукционному предположению $C_p \overset{*}{\underset{G_2}{\Rightarrow}} w_p$. Поэтому $A_i \underset{G_2}{\Rightarrow} u_1u_2 \dots u_r \overset{*}{\underset{G_2}{\Rightarrow}} w_1w_2 \dots w_r$.

Итак, из $A_i \overset{*}{\underset{G_1}{\Rightarrow}} w$ следует вывод $A_i \overset{*}{\underset{G_2}{\Rightarrow}} w$. Поскольку $S \in \{A_1, A_2, \dots, A_k\}$, то $L(G_1) \subseteq L(G_2)$.

II. $L(G_2) \subseteq L(G_1)$. Пусть $\alpha \underset{G_2}{\Rightarrow} \beta$. Покажем, что $\alpha \overset{*}{\underset{G_1}{\Rightarrow}} \beta$. Шаг вывода $\alpha \underset{G_2}{\Rightarrow} \beta$ предполагает применение правила вида $A_i \to u_1u_2 \dots u_r \in P_2$. Его существование обусловлено существованием правила $A_i \to C_1C_2 \dots C_r \in P_1$, такого что либо $C_i = u_i$, если $u_i \in V_T$, либо C_i — нетерминал типа B и $C_i \overset{*}{\underset{G_1}{\Rightarrow}} u_i$ ($i = 1, 2, \dots, r$). Следовательно, $A_i \underset{G_1}{\Rightarrow} C_1C_2 \dots C_r \overset{*}{\underset{G_1}{\Rightarrow}} u_1u_2 \dots u_r$.

Таким образом, применение одного правила $A_i \to u_1u_2 \dots u_r \in P_2$ в выводе $\alpha \underset{G_2}{\Rightarrow} \beta$ равносильно применению нескольких правил из множества P_1, позволяющих в цепочке α заменить A_i на $u_1u_2 \dots u_r$, что дает β. Итак, каждый шаг вывода терминальной цепочки в грамматике G_2 может быть заменен несколькими шагами вывода в грамматике G_1, т.е. $L(G_2) \subseteq L(G_1)$.

Из рассуждений I и II следует, что $L(G_1) = L(G_2)$.

Пример 4.3. Рассмотрим грамматику $G_1 = (\{S, A, B\}, \{a, b, c, d\}, P_1, S)$, где $P_1 = \{S \to ASB, S \to AB, A \to a, A \to b, B \to c, B \to d\}$.

Легко видеть, что A порождает только цепочки a и b, а B порождает только цепочки c и d. Однако, S порождает бесконечно много цепочек.

Правило $S \to ASB$ заменяется правилами $S \to aSc$, $S \to aSd$, $S \to bSc$, $S \to bSd$. Аналогично, правило $S \to AB$ заменяется правилами $S \to ac$, $S \to ad$, $S \to bc$, $S \to bd$.

Новая грамматика есть $G_2 = (\{S\}, \{a, b, c, d\}, P_2, S)$, где

$$P_2 = \{S \to aSc, S \to aSd, S \to bSc, S \to bSd, S \to ac, S \to ad, S \to bc, S \to bd\}.$$

§ 4.5. Свойство самовставленности

***Определение* 4.5.** Говорят, что контекстно-свободная грамматика G является *самовставленной*, если существует нетерминал A, такой, что $A \overset{*}{\underset{G}{\Rightarrow}} \alpha_1 A \alpha_2$, где $\alpha_1, \alpha_2 \in V^+$. Говорят также, что нетерминал A является *самовставленным*.

Заметим, что именно свойство самовставленности является причиной появления цепочек вида uv^iwx^iy. Возможно, некоторые понимают, что это свойство самовставленности отличает строго контекстно-свободные языки от регулярных множеств. Но отметим и то, что просто из-за свойства самовставленности грамматики порождаемый ею язык не обязан быть регулярным.

Например, грамматика $G = (\{S\},\ \{a,\ b\},\ P,\ S)$, где $P = \{S \to aSa,\ S \to aS,\ S \to bS,\ S \to a,\ S \to b\}$, порождает регулярное множество. Действительно, $L(G) = \{a, b\}^+$.

В этом параграфе будет показано, что контекстно-свободная грамматика, которая не является самовставленной, порождает регулярное множество. Следовательно, контекстно-свободный язык не регулярен тогда и только тогда, когда *все* его грамматики — самовставленные.

Теорема 4.10. *Пусть G — несамовставленная контекстно-свободная грамматика. Тогда $L(G)$ — регулярное множество.*

Доказательство. Нетрудно убедиться в том, что если исходная грамматика не является самовставленной, то и эквивалентная ей грамматика в нормальной форме — тоже несамовставленная. В частности, это так для нормальной формы Грейбах. Поэтому, если G — несамовставленная грамматика, то мы можем найти грамматику $G_1 = (V_N^1, V_T, P_1, S_1)$ в нормальной форме Грейбах, эквивалентную грамматике G, которая тоже будет несамовставленной. Хотя это утверждение не очевидно, его легко доказать. Ясно, что применение подстановок, описанных в лемме 4.2 не вводит самовставленности. Что касается преобразований по исключению левой рекурсии, описанных в лемме 4.3, то следует доказать, что нетерминал Z самовставлен, только если нетерминал A — самовставлен. Кроме того, по теореме 4.2 терминальная цепочка может быть выведена из каждого нетерминала.

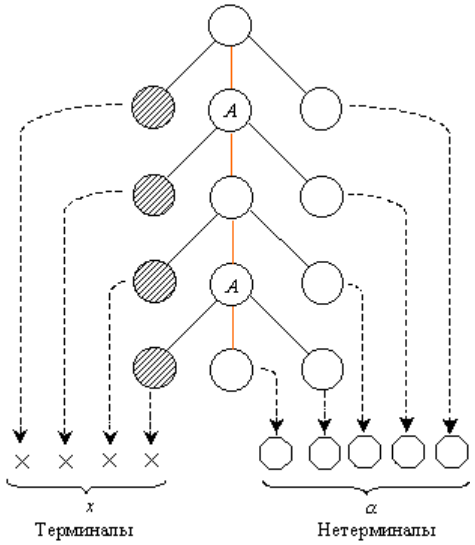

Рис. 4.4.

Рассмотрим левосторонний вывод в грамматике G_1 некоторой сентенциальной формы $x\alpha$. Если G_1 имеет m нетерминалов и l — длина самой длинной правой части правил, то никакая сентенциальная форма не может иметь больше, чем ml нетерминалов. Чтобы убедиться в этом, предположим, что в некоторой сентенциальной форме α левостороннего вывода появляется больше, чем ml нетерминалов. В дереве вывода α рассмотрим путь от корня к крайнему левому листу, помеченному нетерминалом (рис. 4.4). Узлы одного уровня представляют правую часть одного правила грамматики, породившего эти узлы. Все узлы, расположенные справа от этого пути, также как и упомянутый лист, еще не раскрывались с помощью правил. Именно они и образуют цепочку α, состоящую из нетерминалов. На каждом уровне таких узлов не больше $l-2$, кроме самого нижнего. На нижнем же уровне их не больше $l-1$.

Предположим, что в нашем дереве вывода k уровней. Тогда на всех уровнях узлов, порождающих α, не больше, чем $(l-2)(k-1)+l-1$. Всего таких узлов на всех k уровнях по предположению больше ml. Следовательно,
$(l-2)(k-1)+l-1 \geq ml+1,\ k \geq (ml-l+2)/(l-2)+1 = ml/(l-2) > ml/l = m$,
это, естественно, предполагает, что $l > 2$.

Итак, уровней в дереве вывода α (длина пути, о котором шла речь) больше m, т.е. по крайней мере их $m+1$. Следовательно, на этом пути найдутся, по крайней мере, два узла, помеченных одним и тем же нетерминалом A. В этом случае существует левосторонний вывод вида $A \overset{*}{\underset{G_1}{\Rightarrow}} zA\beta$, где $z \in V_T^+$, $\beta \in V_N^{1+}$, т.е. $z \neq \varepsilon$, $\beta \neq \varepsilon$ ($l > 2$). А это значит, что A — самовставленный нетерминал, что противоречит условию теоремы.

Теперь, если в любой сентенциальной форме самое большее ml нетерминалов, мы можем построить грамматику типа 3: $G_2 = (V_N^2,\ V_T,\ P_2,\ S_2)$, порождающую язык $L(G)$ следующим образом. Нетерминалы грамматики G_2 соответствуют цепочкам нетерминалов грамматики G_1, длина которых не больше ml, т.е. $V_N^2 = \{[\alpha] \mid \alpha \in V_N^{1+},\ |\alpha| \leq ml\}$. При этом $S_2 = [S]$. Если $A \to b\alpha \in P_1$, то для всех нетерминалов из словаря V_N^2, соответствующих строкам, начинающимся на A, в множество правил P_2 мы включаем правила вида $[A\beta] \to b[\alpha\beta]$ при условии, что $|\alpha\beta| \leq ml$.

Из построения должно быть очевидно, что грамматика G_2 моделирует все левосторонние выводы в грамматике G_1, так что $L(G_2) = L(G_1)$. Действительно, индукцией по длине вывода легко показать, что $S \overset{*}{\underset{G_1}{\Rightarrow}} x\alpha$ посредством левостороннего вывода тогда и только тогда, когда $[S] \overset{*}{\underset{G_2}{\Rightarrow}} x[\alpha]$. Здесь $x \in V_T^+$ — закрытая, а $\alpha \in V_N^{1*}$ — открытая часть данной сентенциальной формы.

I. Докажем сначала, что если $S \overset{l}{\underset{G_1}{\Rightarrow}} x\alpha$, то $[S] \overset{*}{\underset{G_2}{\Rightarrow}} x[\alpha]$.

База. Пусть $l = 1$. Имеем $S \underset{G_1}{\Rightarrow} x\alpha$, $S \to x\alpha \in P_1$, $x \in V_T$, $\alpha \in V_N^{1*}$. Следовательно, существует правило $[S] \to x[\alpha] \in P_2$ и потому $[S] \underset{G_2}{\Rightarrow} x[\alpha]$.

Индукционная гипотеза. Предположим, что аналогичное утверждение имеет место при всех $l \le n$ $(n \ge 1)$.

Индукционный переход. Докажем утверждение при $l \le n + 1$. Пусть $S \overset{n}{\underset{G_1}{\Rightarrow}} x'A\beta \underset{G_1}{\Rightarrow} x'b\alpha'\beta = x\alpha$, т.е. $x = x'b$, $\alpha = x'\beta$. По индукционной гипотезе из существования вывода $S \overset{n}{\underset{G_1}{\Rightarrow}} x'A\beta$ следует, что $[S] \overset{*}{\underset{G_2}{\Rightarrow}} x'[A\beta]$, а поскольку на последнем шаге вывода использовано правило $A \to b\alpha' \in P_1$, то существует правило $[A\beta] \to b[\alpha'\beta] \in P_2$, с помощью которого можно завершить имеющийся вывод $[S] \overset{*}{\underset{G_2}{\Rightarrow}} x'[A\beta] \underset{G_2}{\Rightarrow} x'b[\alpha'\beta] = x[\alpha]$.

II. Докажем теперь, что если $[S] \overset{l}{\underset{G_2}{\Rightarrow}} x[\alpha]$, то $S \overset{*}{\underset{G_1}{\Rightarrow}} x\alpha$.

База. Пусть $l = 1$. Имеем $[S] \underset{G_2}{\Rightarrow} x[\alpha]$. Существует $[S] \to x[\alpha] \in P_2$, $x \in V_T$, $\alpha \in V_N^{1*}$, которое обусловлено существованием правила $S \to x\alpha \in P_1$, и потому $S \underset{G_1}{\Rightarrow} x\alpha$.

Индукционная гипотеза. Предположим, что аналогичное утверждение имеет место при всех $l \le n$ $(n \ge 1)$.

Индукционный переход. Докажем утверждение при $l \le n + 1$.

Пусть $[S] \overset{n}{\underset{G_2}{\Rightarrow}} x'[A\beta] \underset{G_2}{\Rightarrow} x'b[\alpha'\beta] = x[\alpha]$. По индукционной гипотезе из существования вывода $[S] \overset{n}{\underset{G_2}{\Rightarrow}} x'[A\beta]$ следует, что $S \overset{*}{\underset{G_1}{\Rightarrow}} x'A\beta$. На последнем шаге вывода использовано правило $[A\beta] \to b[\alpha'\beta] \in P_2$, существование которого обусловлено существованием правила $A \to b\alpha' \in P_1$, которое можно использовать для завершения имеющегося вывода $S \overset{*}{\underset{G_1}{\Rightarrow}} x'A\beta \underset{G_1}{\Rightarrow} x'b\alpha'\beta = x\alpha$.

Из рассуждений I и II при $\alpha = \varepsilon$ получаем $L(G_1) = L(G_2)$. Таким образом, язык $L(G)$ — регулярен. Что и требовалось доказать.

§ 4.6. ε-Правила в контекстно-свободных грамматиках

Ранее мы показали, что на правила КС-грамматик можно накладывать некоторые *ограничения*, не сужая класс языков, которые могут порождаться. Теперь мы рассмотрим *расширения* КС-грамматик, которые разрешают использовать правила вида $A \to \varepsilon$ для *любого* нетерминала. Такое правило называется ε-*правилом* или ε-*порождением*. Многие описания синтаксиса языков программирования допускают такие порождения. Мы покажем, что язык, порождаемый КС-грамматикой с ε-правилами, — всегда КС-язык.

Понятия, касающиеся деревьев вывода для КС-грамматик, непосредственно переносятся на эти расширенные грамматики. Просто разрешается использовать обозначение ε в качестве метки узла. Ясно, что такой узел должен быть листом.

Теорема 4.11. *Если L — язык, порождаемый грамматикой $G = (V_N, V_T, P, S)$, и каждое правило в P имеет вид $A \to \alpha$, где A — нетерминал, а $\alpha \in V^*$ ($\alpha = \varepsilon$ допустимо), то L может быть порожден грамматикой, в которой каждое правило имеет вид $A \to \alpha$, где A — нетерминал, а $\alpha \in V^+$, либо $S \to \varepsilon$ и, кроме того, начальный нетерминал грамматики S не появляется в правой части никакого правила.*

Доказательство. При помощи тривиального расширения леммы 2.1 мы можем предположить, что S не появляется справа ни в одном правиле в P. Для любого нетерминала $A \in V_N$ мы можем решить, существует ли вывод $A \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$, поскольку если такой вывод существует, то существует и дерево вывода, ветви которого не длиннее, чем число нетерминалов грамматики G (этот аргумент использовался в теореме 4.1).

Пусть $A_1, A_2, \ldots, A_k$ — те нетерминалы из словаря V_N, из которых цепочка ε может быть выведена, а $B_1, B_2, \ldots, B_m$ — те нетерминалы, из которых цепочка ε не выводима. Мы построим новое множество правил P_1 следующим образом.

Если $S \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$, то в P_1 включим правило $S \to \varepsilon$. Никакие правила вида $A \to \varepsilon$ в P_1 не включаются.

Если $A \to C_1C_2 \ldots C_r \in P$, $r \geq 1$, то в P_1 включаются правила вида $A \to \alpha_1\alpha_2 \ldots \alpha_r$, где $\alpha_i = C_i$, если $C_i \in V_T \cup \{B_1, B_2, \ldots, B_m\}$, либо $\alpha_i = C_i$ или $\alpha_i = \varepsilon$, если $C_i \in \{A_1, A_2, \ldots, A_k\}$, однако *не все* $\alpha_i = \varepsilon$. Другими словами, преобразования на шаге 3 состоят в том, что в правой части A-правила каждое вхождение A альтернативно либо подменяется на ε, либо остается, как есть. Вхождения других символов не затрагиваются. При этом не допускается, чтобы правая часть обратилась в ε.

Ясно, что новая грамматика $G_1 = (V_N, V_T, P_1, S)$ отличается от грамматики G только набором правил, причем все они имеют требуемый вид.

I. Докажем, что $L(G_1) \subseteq L(G)$. Пусть $\alpha \underset{G_1}{\Rightarrow} \beta$ и при этом применяется правило $A \to \alpha_1\alpha_2 \ldots \alpha_r \in P_1$. Его применение эквивалентно применению правила $A \to C_1C_2 \ldots C_r \in P$, из которого оно было получено, и нескольких правил из множества правил P для выводов $C_i \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$, если $\alpha_i = \varepsilon$.

II. Докажем теперь, что $L(G) \subseteq L(G_1)$. Индукцией по числу шагов l в выводе докажем, что если $A \overset{l}{\underset{G}{\Rightarrow}} w$ и $w \neq \varepsilon$, то $A \overset{*}{\underset{G_1}{\Rightarrow}} w$ для $A \in V_N$.

База. Пусть $l = 1$. Очевидно, что вывод $A \underset{G}{\Rightarrow} w$ есть также вывод $A \underset{G_1}{\Rightarrow} w$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех выводов длиной $l \leq n$ ($n \geq 1$).

Индукционный переход. Пусть $A \overset{n+1}{\underset{G}{\Rightarrow}} w$. Более детально этот вывод имеет следующий вид: $A \underset{G}{\Rightarrow} C_1C_2 \ldots C_r \overset{n}{\underset{G}{\Rightarrow}} w_1w_2 \ldots w_r$, причем $C_i \overset{l_i}{\underset{G}{\Rightarrow}} w_i$, $l_i \leq n$. Если $w_i \neq \varepsilon$, то по индукционному предположению $C_i \overset{*}{\underset{G_1}{\Rightarrow}} w_i$. Кроме того, по постро-

нию из правила $A \to C_1 C_2 \ldots C_r \in P$ получается правило $A \to \alpha_1 \alpha_2 \ldots \alpha_r \in P_1$, где $\alpha_i = C_i$, если $w_i \neq \varepsilon$, или $\alpha_i = \varepsilon$, если $w_i = \varepsilon$. Следовательно, $A \overset{*}{\underset{G_1}{\Rightarrow}} w$.

Из рассуждений I и II следует, что $L(G_1) = L(G)$. Что и требовалось доказать.

Из теоремы 4.11 непосредственно следует тот факт, что единственная разница между контекстно-свободной грамматикой с правилами вида $A \to \varepsilon$ и грамматиками без ε-правил состоит в том, что первая может порождать пустое предложение. Далее мы будем называть cfg с ε-правилами просто cfg, зная, что эквивалентная грамматика без ε-правил (за исключением быть может $S \to \varepsilon$) может быть найдена.

§ 4.7. Специальные типы контекстно-свободных языков и грамматик

Здесь мы рассмотрим несколько ограниченных классов КС-языков.

***Определение* 4.6.** Говорят, что контекстно-свободная грамматика $G = (V_N, V_T, P, S)$ — *линейна*, если каждое ее правило имеет вид $A \to uBv$ или $A \to u$, где $A, B \in V_N$, $u, v \in V_T^*$. Если $v = \varepsilon$, то грамматика называется *праволинейной*, если $u = \varepsilon$, то она *леволинейна*.

Язык, который может порождаться линейной грамматикой, называется *линейным* языком.

Не все контекстно-свободные языки являются линейными языками. Заметим, что ни одна цепочка, выводимая в линейной грамматике, не имеет более одного нетерминала.

Пример 4.4. Грамматика $G = (\{S\}, \{0, 1\}, P, S)$, где $P = \{S \to 0S1, S \to \varepsilon\}$, является линейной грамматикой, которая порождает язык $L = \{0^n 1^n \mid n \geq 0\}$.

***Определение* 4.7.** Говорят, что грамматика $G = (V_N, V_T, P, S)$ — *последовательная*, если нетерминалы $A_1, A_2, \ldots, A_k$ из словаря V_N можно упорядочить так, что если $A_i \to \alpha \in P$, то α не содержит ни одного нетерминала A_j с индексом $j < i$.

Язык, порождаемый последовательной грамматикой, называется *последовательным языком*.

Пример 4.5. Грамматика $G = (\{A_1, A_2\}, \{0, 1\}, P, A_1)$, где $P = \{A_1 \to A_2 A_1, A_1 \to A_2, A_2 \to 0A_2 1, A_2 \to \varepsilon\}$, является последовательной грамматикой, которая порождает язык $L = \{0^n 1^n \mid n \geq 0\}^*$.

***Определение* 4.8.** Если контекстно-свободный язык L над алфавитом V_T есть подмножество языка $w_1^* w_2^* \ldots w_k^*$[6] для некоторого k, где $w_i \in V_T^*$, $i = 1, 2, \ldots, k$, то говорят, что L — *ограниченный язык*.

---

[6] Строго говоря, $w_1^* w_2^* \ldots w_k^*$ следовало бы записывать в виде $\{w_1\}^* \{w_2\}^* \ldots \{w_k\}^*$. Но и без скобок не возникает никаких недоразумений.

Пример 4.6. Язык $\{(ab)^n c^n (dd)^* \mid n \geq 1\}$ является ограниченным языком. Здесь $k = 3$, а $w_1 = ab$, $w_2 = c$, $w_3 = d$.

***Определение* 4.9.** Говорят, что контекстно-свободная грамматика $G = (V_N, V_T, P, S)$ — *неоднозначна*, если в языке $L(G)$ существует цепочка с двумя или более различными левосторонними выводами.

Если все грамматики, порождающие некоторый контекстно-свободный язык, неоднозначны, то говорят, что этот язык существенно неоднозначен.

Существенно неоднозначные контекстно-свободные языки существуют. Классическим примером такого языка является язык $L = \{a^i b^j c^k \mid i = j \text{ или } j = k\}$. Основная причина, по которой язык L существенно неоднозначен, состоит в том, что любая cfg, порождающая язык L, должна порождать те цепочки, для которых $i = j$, при помощи процесса, который отличается от процесса порождения тех цепочек, для которых $j = k$. Невозможно не порождать некоторые из тех цепочек, для которых $i = j = k$, посредством обоих процессов. Строгое доказательство этого факта весьма сложно (см., например, [1]).

Известно, что проблема распознавания существенной неоднозначности КС-языков алгоритмически неразрешима.

Пример 4.7. Рассмотрим грамматику G из примера 4.1, которая имеет следующие правила:

$$P = \{S \to bA, \quad S \to aB,$$
$$A \to a, \quad A \to aS, \quad A \to bAA,$$
$$B \to b, \quad B \to bS, \quad B \to aBB\}.$$

Цепочка *aabbab* имеет следующие два левосторонних вывода:

$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabB \Rightarrow aabbS \Rightarrow aabbaB \Rightarrow aabbab$,

$S \Rightarrow aB \Rightarrow aaBB \Rightarrow aabSB \Rightarrow aabbAB \Rightarrow aabbaB \Rightarrow aabbab$.

Следовательно, грамматика G — неоднозначная. Однако язык состоит из цепочек, содержащих равное число букв a и b, и может быть порожден однозначной грамматикой $G_1 = (\{S, A, B\}, \{a, b\}, P, S)$, где P состоит из правил

$$S \to aBS, S \to aB, S \to bAS, S \to bA, A \to bAA, A \to a, B \to aBB, B \to b.$$

Глава 5

МАГАЗИННЫЕ АВТОМАТЫ

§ 5.1. Неформальное описание

В этой главе мы рассмотрим простое устройство — магазинный автомат[7] (pda — pushdown automaton), которое адекватно классу КС-языков в том смысле, что любой КС-язык распознается каким-нибудь магазинным автоматом, и любой магазинный автомат распознает некоторый КС-язык.

Магазинный автомат подобен конечному автомату, но в отличие от последнего имеет рабочую память — *магазин*, в который записываются символы из еще одного алфавита — *алфавита магазинных символов*. Каждое движение магазинного автомата определяется в зависимости от текущего *состояния управления*, *входного символа* или независимо от него — так называемые ε-*движения* и от *верхнего символа магазина*. Одно движение магазинного автомата состоит в замещении верхнего символа магазина некоторой магазинной цепочкой, в частности пустой (стирание верхнего символа магазина), и переходе в новое состояние управления. При этом текущим входным символом становится следующий символ на входной ленте, если выполняется движение, зависящее от входного символа, либо текущий входной символ остается тем же самым, если выполняется ε-движение. Поскольку движение зависит от верхнего символа магазина, то с самого начала в магазине находится один символ — *начальный символ магазина*.

Считается, что некоторая цепочка *принята*, если магазинный автомат из начального состояния управления, имея в магазине единственный — начальный символ магазина и прочитав данную цепочку на входе, переходит в одно из своих *конечных состояний* или *опустошает магазин*. Каждый конкретный магазинный автомат использует только какой-нибудь один из этих двух признаков приема входной цепочки. Как и в случае конечных автоматов существуют две модели магазинных автоматов — *недетерминированная* и *детерминированная*. В недетерминированной модели автомат каждый раз имеет возможность некоторого конечного выбора движений и совершает одно из них. Входная цепочка считается принятой, если существует хотя бы одна последовательность выборов движений, которая приводит автомат к *конечной конфигурации* — входная цепочка прочитана, текущее состояние — конечное или (в другом варианте) магазин пуст. В противном случае она не принимается. Множество всех цепочек, принимаемых данным магазинным автоматом, называется *языком*, *распознаваемым этим магазинным автоматом*.

[7] Вместо этого термина часто используется сокращение МП-автомат.

В данной главе будет показано, что оба определения приема эквивалентны в том смысле, что если язык принимается некоторым магазинным автоматом при пустом магазине, то он может быть принят некоторым другим магазинным автоматом при конечном состоянии, и наоборот.

Кроме того, будет доказана основная теорема о том, что язык принимается недетерминированным МП-автоматом тогда и только тогда, когда он является КС-языком. Известно, что класс языков, принимаемых детерминированными МП-автоматами, является строгим подклассом языков, принимаемых недетерминированными МП-автоматами.

§ 5.2. Формальное определение

***Определение* 5.1.** Недетерминированный магазинный автомат есть *формальная система* $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, где Q — конечное *множество состояний*; Σ — конечный *входной алфавит*; Γ — конечный *магазинный алфавит;* δ — *отображение типа* $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$, представляющее конечное *управление автомата;* $q_0 \in Q$ *— начальное состояние*; $Z_0 \in \Gamma$ *— начальный символ магазина*, который в самом начале является единственным содержимым магазина; $F \subseteq Q$ *— множество конечных состояний.*

Мы будем придерживаться следующей системы обозначений: строчные буквы из начала латинского алфавита — отдельные входные символы; строчные буквы из конца латинского алфавита служат для обозначения цепочек входных символов; строчные греческие буквы — цепочки магазинных символов; прописные латинские буквы — отдельные магазинные символы.

***Определение* 5.2.** Для описания движений МП-автомата будем использовать понятие *конфигурации*, под которой будем подразумевать тройку (q, x, α), где $q \in Q$ *— текущее состояние управления*; $x \in \Sigma^*$ *— непросмотренная часть входной цепочки* (от текущего символа до ее конца) ; $\alpha \in \Gamma^*$ — *магазинная цепочка*, причем крайний левый ее символ считается находящимся на вершине магазина.

Начальная конфигурация есть (q_0, x, Z_0), где x — вся входная цепочка. *Конечная конфигурация* определяется по-разному: в зависимости от того, какой признак приема используется. Если прием определяется при конечном состоянии, то конечная конфигурация есть (q, ε, α) , где $q \in F$ — автомат достиг конечного состояния; ε означает, что вся входная цепочка прочитана; $\alpha \in \Gamma^*$ — произвольная магазинная цепочка. Достижение конечного состояния не означает завершения работы автомата, а сигнализирует лишь о том, что прочитанная к этому моменту часть входной цепочки принимается. За время сканирования входной цепочки конечные состояния могут достигаться несколько раз. Если прием определяется при пустом магазине, то конечная конфигурация есть $(q, \varepsilon, \varepsilon)$, где $q \in Q$ — любое состояние; ε во второй позиции означает, как и в предыдущем случае,

что вся входная цепочка прочитана; ε в третьей позиции означает, что магазин пуст. При пустом магазине никакое дальнейшее движение не определено.

Запись $\delta(q, a, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_m, \gamma_m)\}$, где $q, p_i \in Q$, $a \in \Sigma$, $Z \in \Gamma$, $\gamma_i \in \Gamma^*$ $(i = 1, 2, \ldots, m)$, означает, что магазинный автомат в состоянии q с символом a на входе и символом Z на вершине магазина может для любого i перейти в состояние p_i, заменить Z на γ_i и продвинуться на входе к следующей позиции (при этом крайний левый символ γ_i окажется на вершине магазина).

Пусть текущая конфигурация имеет вид: $(q, ax, Z\alpha)$, где $x \in \Sigma^*$. В терминах конфигураций одно из возможных движений представляется следующим образом: $(q, ax, Z\alpha) \vdash (p_i, x, \gamma_i \alpha)$ для любого i, $1 \le i \le m$.

Запись $\delta(q, \varepsilon, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_m, \gamma_m)\}$ означает, что pda в состоянии q независимо от того, какой символ на входе, и с символом Z на вершине магазина может для любого i перейти в состояние p_i, заменить Z на γ_i, не изменяя текущей позиции на входе.

Пусть текущая конфигурация имеет вид $(q, x, Z\alpha)$. Тогда в терминах конфигураций одно из возможных движений представляется следующим образом: $(q, x, Z\alpha) \vdash (p_i, x, \gamma_i \alpha)$ для любого i, $1 \le i \le m$.

Если $\gamma_i = \varepsilon$, то происходит стирание верхнего символа магазина — вершина магазина опускается; если $|\gamma_i| = 1$, то происходит замена верхнего символа магазина; если $|\gamma_i| > 1$, то вершина магазина поднимается.

Таким образом, мы ввели на множестве конфигураций pda отношение непосредственного следования одной конфигурации за другой, использовав для него символ $\vdash$.

Естественно ввести степень отношения $\vdash^n$, транзитивное замыкание $\vdash^+$ и рефлексивно-транзитивное замыкание $\vdash^*$ этого отношения на множестве конфигураций. При необходимости явно показать, движения какого pda рассматриваются, под соответствующими значками указывается имя автомата, например: $\vdash_M$, $\vdash_M^n$, $\vdash_M^+$ или $\vdash_M^*$. Напомним, что содержательно эти значки обозначают переход от конфигурации, стоящей слева от значка, к конфигурации, стоящей справа от значка, соответственно за одно движение, за n движений, за положительное число движений, за произвольное, может быть нулевое, число движений.

***Определение* 5.3.** *Язык, принимаемый магазинным автоматом* $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ *при конечном состоянии*, определим как множество $T(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \alpha), q \in F\}$.

***Определение* 5.4.** *Язык, принимаемый магазинным автоматом* $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ *при пустом магазине*, определим как множество $N(M) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon), q \in Q\}$. В этом случае неважно, каким является множество конечных состояний, и часто оно просто считается пустым.

Пример 5.1. Пусть pda $M = (\{q_1, q_2\}, \{0, 1\}, \{R, B, G\}, \delta, q_1, R, \varnothing)$, где

1) $\delta(q_1, 0, R) = \{(q_1, BR)\}$,
2) $\delta(q_1, 1, R) = \{(q_1, GR)\}$,
3) $\delta(q_1, 0, B) = \{(q_1, BB), (q_2, \varepsilon)\}$,
4) $\delta(q_1, 0, G) = \{(q_1, BG)\}$,
5) $\delta(q_1, 1, B) = \{(q_1, GB)\}$,
6) $\delta(q_1, 1, G) = \{(q_1, GG), (q_2, \varepsilon)\}$,
7) $\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$,
8) $\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$,
9) $\delta(q_1, \varepsilon, R) = \{(q_2, \varepsilon)\}$,
10) $\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$;

M — недетерминированный магазинный автомат, принимающий язык $N(M) = \{ww^R \mid w \in \{0, 1\}^*\}$, где w^R обозначает инвертированную цепочку w, при пустом магазине.

Правила 1–6 позволяют pda M запомнить входной символ в магазинной памяти. При этом входной символ 0 отображается в магазине посредством символа B, а символ 1 отображается в магазине посредством символа G.

Правила 3 и 6 предоставляют автомату выбор движений между запоминанием очередного входного символа в магазине (в предположении, что середина входной цепочки не достигнута) и переходом в режим сопоставления текущего входного символа с символом на вершине магазина и стиранием последнего в случае их соответствия (в предположении, что достигнута середина входной цепочки).

Если на входе находится цепочка вида ww^R, то существует такая последовательность движений, которая опустошает магазин к моменту достижения конца цепочки. Например, если на вход поступает цепочка 0110, то автомат имеет выбор движений, представленный на рис. 5.1, где приведено дерево возможных переходов между конфигурациями. Существующая среди них следующая последовательность движений:

$$(q_1, 0110, R) \vdash (q_1, 110, BR) \vdash (q_1, 10, GBR) \vdash (q_2, 0, BR) \vdash (q_2, \varepsilon, R) \vdash (q_2, \varepsilon, \varepsilon)$$

дает основание заключить, что входная цепочка 0110 принимается.

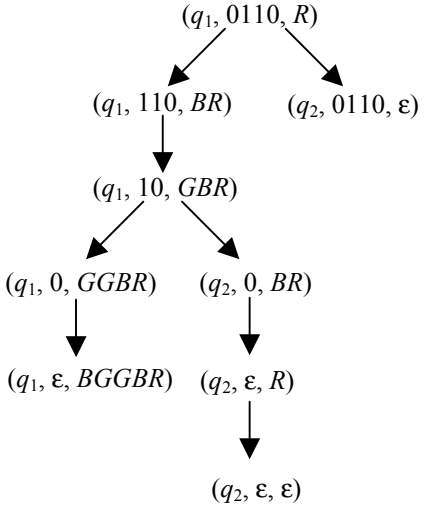

Рис. 5.1.

Пример 5.2. Пусть pda $M = (\{q_1, q_2\}, \{0, 1, c\}, \{R, B, G\}, \delta, q_1, R, \varnothing)$, где

1) $\delta(q_1, 0, R) = \{(q_1, BR)\}$,
2) $\delta(q_1, 0, B) = \{(q_1, BB)\}$,
3) $\delta(q_1, 0, G) = \{(q_1, BG)\}$,
4) $\delta(q_1, c, R) = \{(q_2, R)\}$,
5) $\delta(q_1, c, B) = \{(q_2, B)\}$,
6) $\delta(q_1, c, G) = \{(q_2, G)\}$,
7) $\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$,
8) $\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$,
9) $\delta(q_2, 1, G) = \{(q_2, \varepsilon)\}$,
10) $\delta(q_1, 1, R) = \{(q_1, GR)\}$,
11) $\delta(q_1, 1, B) = \{(q_1, GB)\}$,
12) $\delta(q_1, 1, G) = \{(q_1, GG)\}$;

Легко сообразить, что $N(M) = \{wcw^R \mid w \in \{0,1\}^*\}$, где w^R обозначает инвертированную цепочку w.

Рассмотрим движения pda M при наличии цепочки $01c10$ на его входе:

$$(q_1, 01c10, R) \underset{M}{\vdash} (q_1, 1c10, BR) \underset{M}{\vdash} (q_1, c10, GBR) \underset{M}{\vdash}$$
$$(q_2, 10, GBR) \underset{M}{\vdash} (q_2, 0, BR) \underset{M}{\vdash} (q_2, \varepsilon, R) \underset{M}{\vdash} (q_2, \varepsilon, \varepsilon).$$

Следовательно, цепочка $01c10$ принимается при пустом магазине.

Заметим, что равенство $\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$ означает движение, не зависящее от входного символа, каким бы он ни был, — в любом случае происходит стирание верхнего символа R магазина без продвижения по входной цепочке и переход в состояние q_2.

Магазинный автомат из примера 5.2 является детерминированным в том смысле, что из любой конфигурации возможно не более одного движения.

***Определение* 5.5.** Магазинный автомат $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ является детерминированным, если

1) для любых $q \in Q$, $Z \in \Gamma$ и $a \in (\Sigma \cup \{\varepsilon\})$ значение $\#\delta(q, a, Z) \le 1$;

2) для любых $q \in Q$ и $Z \in \Gamma$ всякий раз, как $\delta(q, \varepsilon, Z) \neq \varnothing$, значение $\delta(q, a, Z) = \varnothing$ для всех $a \in \Sigma$.

Условие 1 означает, что если движение определено, то оно единственно. Условие 2 предотвращает выбор между ε-движением и движением, использующим входной символ.

Для конечных автоматов детерминированная и недетерминированная модели эквивалентны по отношению к применяемым языкам (см. теорему 3.4). Далее мы увидим, что это не так для МП-автоматов. Действительно, язык, состоящий из цепочек вида ww^R, принимается недетерминированным pda, но не существует никакого детерминированного pda, который принимал бы такой язык.

§ 5.3. Недетерминированные магазинные автоматы и контекстно-свободные языки

Теперь докажем фундаментальный результат: класс языков, принимаемых недетерминированными МП-автоматами, есть в точности класс контекстно-свободных языков. Для этого сначала покажем, что языки, принимаемые недетерминированными МП-автоматами при конечном состоянии, являются в точ-

ности языками, принимаемыми недетерминированными МП-автоматами при пустом магазине, а затем, что языки, принимаемые при пустом магазине, совпадают с классом КС-языков.

Теорема 5.1. *Язык $L = N(M_1)$ для некоторого недетерминированного магазинного автомата M_1 тогда и только тогда, когда $L = T(M_2)$ для некоторого магазинного автомата M_2.*

Доказательство.

I. *Необходимость*. Пусть $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \varnothing)$ — недетерминированный pda, такой, что $L = N(M_1)$. Положим

$$M_2 = (Q \cup \{q_0', q_f\}, \Sigma, \Gamma \cup \{X\}, \delta', q_0', X, \{q_f\}),$$

где δ' определяется следующим образом:

1) $\delta'(q_0', \varepsilon, X) = \{(q_0, Z_0X)\}$;

2) $\delta'(q, a, Z) = \delta(q, a, Z)$ для всех $q \in Q$, $a \in (\Sigma \cup \{\varepsilon\})$ и $Z \in \Gamma$;

3) $\delta'(q, \varepsilon, X) = \{(q_f, \varepsilon)\}$ для всех $q \in Q$.

Правило 1 воспроизводит начальную конфигурацию автомата M_1. При этом начальный символ магазина X остается в качестве своеобразного маркера дна магазина до тех пор, пока он не будет удален последним движением автомата M_2. Выше этого маркера при последующих движениях, повторяющих движения pda M_1, всегда будут находиться только магазинные символы автомата M_1. Это обеспечивается правилом 2. Когда же будет воспроизведено последнее движение pda M_1, опустошающее его магазин, на вершине магазина автомата M_2 появится маркер дна. В этот момент автомат M_2 посредством правила 3 переводится в конечное состояние, тем самым сигнализируя прием входной цепочки.

Заметим, что движение по правилу 3 стирает символ магазина X, но могло бы записать вместо него любую магазинную цепочку.

I.1. Докажем, сначала, что если $x \in N(M_1)$, то $x \in T(M_2)$.

Пусть $x \in N(M_1)$, т.е. $(q_0, x, Z_0) \overset{*}{\underset{M_1}{\vdash}} (q, \varepsilon, \varepsilon)$. Посмотрим, какие движения будет совершать автомат M_2, имея на входе цепочку символов x. Согласно правилу 1 $(q_0', x, X) \underset{M_2}{\vdash} (q_0, x, Z_0X)$. Далее согласно правилу 2 автомат M_2, повторяя движения автомата M_1, осуществляет переход $(q_0, x, Z_0X) \overset{*}{\underset{M_2}{\vdash}} (q, \varepsilon, X)$. Наконец, по правилу 3 имеем последнее движение: $(q, \varepsilon, X) \underset{M_2}{\vdash} (q_f, \varepsilon, \varepsilon)$. Таким образом, $x \in T(M_2)$.

I.2. Теперь докажем, что если $x \in T(M_2)$, то $x \in N(M_1)$.

Пусть $x \in T(M_2)$, т.е. $(q_0', x, X) \overset{*}{\underset{M_2}{\vdash}} (q_f, \varepsilon, \gamma)$ для некоторой цепочки $\gamma \in (\Gamma \cup \{X\})^*$, причем по построению автомата M_2 $\gamma = \varepsilon$. Выделив здесь первый шаг явным образом, имеем $(q_0', x, X) \underset{M_2}{\vdash} (q_0, x, Z_0X) \overset{*}{\underset{M_2}{\vdash}} (q_f, \varepsilon, \varepsilon)$. Заключительная конфигурация $(q_f, \varepsilon, \varepsilon)$ достижима лишь посредством ε-движения благодаря прави-

лу 3 в случае, когда на вершине магазина находится X. Следовательно, предпоследняя конфигурация должна иметь вид (q_0, ε, X). Итак, все движения выстраиваются в следующую последовательность конфигураций:

$$(q_0', x, X) \vdash_{M_2} (q_0, x, Z_0X) \overset{*}{\vdash}_{M_2} (q, \varepsilon, X) \vdash_{M_2} (q_f, \varepsilon, \varepsilon).$$

Но на главном участке $(q_0, x, Z_0X) \overset{*}{\vdash}_{M_2} (q, \varepsilon, X)$ автомат M_2 просто повторяет движения автомата M_1. Поэтому $(q_0, x, Z_0) \overset{*}{\vdash}_{M_1} (q, \varepsilon, \varepsilon)$ и $x \in N(M_1)$.

Из рассуждений, проведенных в пп.I.1 и I.2 следует, что $T(M_2) = N(M_1)$.

II. *Достаточность*. Пусть $M_2 = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ — недетерминированный pda, такой, что $L = T(M_2)$. Положим

$$M_1 = (Q \cup \{q_0', q_e\}, \Sigma, \Gamma \cup \{X\}, \delta', q_0', X, \varnothing),$$

где δ' определяется следующим образом:

1) $\delta'(q_0', \varepsilon, X) = \{(q_0, Z_0X)\}$;

2) $\delta'(q, a, Z) = \delta(q, a, Z)$ для всех $q \in Q$, $a \in (\Sigma \cup \{\varepsilon\})$ и $Z \in \Gamma$;

3) $\delta'(q, \varepsilon, Z) = \{(q_e, \varepsilon)\}$ для всех $q \in F$ и $Z \in \Gamma \cup \{X\}$;

4) $\delta'(q_e, \varepsilon, Z) = \{(q_e, \varepsilon)\}$ для всех $Z \in \Gamma \cup \{X\}$.

Правило 1 воспроизводит начальную конфигурацию автомата M_2. При этом начальный символ магазина X остается в качестве своеобразного маркера дна магазина до тех пор, пока он не будет удален последним движением автомата M_1. Выше этого маркера при последующих движениях, повторяющих движения автомата M_2, всегда будут находиться только магазинные символы автомата M_2. Это обеспечивается правилом 2. Если когда-либо pda M_2 входит в конечное состояние, то правила 3 и 4 позволяют pda M_1 выбирать переход в состояние q_e и, не продвигаясь по входу, очищать магазин, тем самым принимая вход или продолжая моделировать движения автомата M_2.

Следует заметить, что автомат M_2 может на неправильном входе опустошить свой магазин, не заходя в конечное состояние и, следовательно, не принимая входную цепочку. Чтобы неправильные цепочки не принимались pda M_1, и введено его собственное особое дно в виде символа X, чтобы он, воспроизводя эти же движения, тоже не опустошил свой магазин и таким образом принял бы неправильную цепочку.

Остается убедиться в том, что $N(M_1) = T(M_2)$.

II.1. Докажем, сначала, что если $x \in N(M_1)$, то $x \in T(M_2)$.

Если $x \in N(M_1)$, то по определению $(q_0', x, X) \overset{*}{\vdash}_{M_1} (q, \varepsilon, \varepsilon)$ при некотором $q \in Q \cup \{q_0', q_e\}$. Рассмотрим подробнее эти движения. Согласно правилу 1 имеем $(q_0', x, X) \vdash_{M_1} (q_0, x, Z_0X)$. Далее $(q_0, x, Z_0X) \overset{*}{\vdash}_{M_1} (q, \varepsilon, \varepsilon)$ при некотором $q \in Q \cup \{q_0', q_e\}$.

Удалить X из магазина согласно построению автомата M_1 возможно только посредством правил 3 или 4, причем правило 4 применимо только после того, как применено правило 3. В свою очередь, правило 3 применимо только в том случае, если автомат M_1 достиг одного из своих конечных состояний. С этого момента происходят только ε-движения, стирающие содержимое магазина. С учетом этого имеем

$$(q_0', x, X) \vdash_{M_1} (q_0, x, Z_0X) \vdash_{M_1}^{*} (q, \varepsilon, \gamma X) \vdash_{M_1} (q_e, \varepsilon, \gamma') \vdash_{M_1}^{*} (q_e, \varepsilon, \varepsilon).$$

Здесь $q \in F$, $\gamma \in \Gamma^*$, $\gamma' \in (\Gamma \cup \{X\})^*$, причем если $\gamma' = \varepsilon$, то перехода $(q_e, \varepsilon, \gamma') \vdash_{M_1}^{*} (q_e, \varepsilon, \varepsilon)$ фактически нет.

Но мы знаем, что согласно правилу 2 автомат M_2 на главном участке просто повторяет движения автомата M_1: $(q_0, x, Z_0X) \vdash_{M_1}^{*} (q, \varepsilon, \gamma X)$, т.е. имеет место переход $(q_0, x, Z_0X) \vdash_{M_2}^{*} (q, \varepsilon, \gamma X)$, где $q \in F$. Следовательно, $x \in T(M_2)$.

II.2. Теперь докажем, что если $x \in T(M_2)$, то $x \in N(M_1)$.

Пусть $x \in T(M_2)$, т.е. $(q_0, x, Z_0) \vdash_{M_2}^{*} (q, \varepsilon, \gamma)$, где $q \in F$, $\gamma \in \Gamma^*$. Посмотрим, как будет действовать автомат M_1 на том же входе. Согласно правилу 1 имеем $(q_0', x, X) \vdash_{M_1} (q_0, x, Z_0X)$. Затем согласно правилу 2 воспроизводятся указанные ранее движения автомата M_2, т.е. имеем $(q_0', x, X) \vdash_{M_1} (q_0, x, Z_0X) \vdash_{M_1}^{*} (q, \varepsilon, \gamma X)$, где $q \in F$, $\gamma \in \Gamma^*$. Далее согласно правилам 3 и 4 имеем $(q, \varepsilon, \gamma X) \vdash_{M_1} (q_e, \varepsilon, \gamma') \vdash_{M_1}^{*} (q_e, \varepsilon, \varepsilon)$, где $\gamma'(\Gamma \cup \{X\})^*$. Если $\gamma' = \varepsilon$, то последнего перехода фактически нет.

Итак, существует последовательность конфигураций:

$$(q_0', x, X) \vdash_{M_1} (q_0, x, Z_0X) \vdash_{M_1}^{*} (q, \varepsilon, \gamma X) \vdash_{M_1} (q_e, \varepsilon, \gamma') \vdash_{M_1}^{*} (q_e, \varepsilon, \varepsilon),$$

означающая, что $x \in N(M_1)$.

Из рассуждений. проведенных в пп. II.1 и II.2 следует, что $N(M_1) = T(M_2)$.

Заключение теоремы следует из доказанных утверждений I и II.

Теорема 5.2. *Если L — контекстно-свободный язык, то существует недетерминированный магазинный автомат M, такой, что $L = N(M)$.*

Доказательство. Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика в нормальной форме Грейбах, и $L = L(G)$. Мы предполагаем, что $\varepsilon \notin L(G)$.

Пусть $S \underset{lm}{\overset{*}{\Rightarrow}} x\alpha$, где $x \in V_T^*$, причем $\alpha = A\beta$, $A \in V_N$, $\beta \in V_N^*$, либо $\alpha = \varepsilon$. Цепочка x называется *закрытой*, а α — *открытой частью* сентенциальной формы $x\alpha$.

Построим недетерминированный pda M таким образом, чтобы в его магазине воспроизводились последовательности сентенциальных форм, составляющих левосторонние выводы в грамматике G, вернее, открытые части таких форм. Для этого положим $M = (\{q\}, V_T, V_N, \delta, q, S, \varnothing)$, где $(q, \gamma) \in \delta(q, a, A)$, если существует правило $A \to a\gamma \in P$. Очевидно, что в магазине pda M всегда находится

нетерминальная цепочка, причем на вершине — символ, подлежащий замене на текущем шаге левостороннего вывода.

Чтобы показать, что $L(G) = N(M)$, достаточно доказать, что $A \overset{*}{\underset{\text{lm}}{\Rightarrow}} x\alpha$ тогда и только тогда, когда $(q, x, A) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, \alpha)$ для любых $x \in V_T^*$, $A \in V_N$, $\alpha \in V_N^*$. Утверждение теоремы последует как частный случай этого вспомогательного утверждения при $A = S$, $\alpha = \varepsilon$: $S \overset{*}{\underset{G}{\Rightarrow}} x$ тогда и только тогда, когда $(q, x, S) \underset{M}{\vdash} (q, \varepsilon, \varepsilon)$.

Заметим, что pda M не совершает ε-движений.

I. Индукцией по длине вывода l покажем, что если $A \overset{l}{\underset{G}{\Rightarrow}} x\alpha$, где $x \in V_T^*$, $\alpha \in V_N^*$, то $(q, x, A) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, \alpha)$.

База. Пусть $l = 1$. Имеем $A \underset{G}{\Rightarrow} x\alpha$. Это значит, что существует правило $A \to x\alpha \in P$, где $x \in V_T$, $\alpha \in V_N^*$. Одновременно по построению автомата M мы имеем $(q, \alpha) \in \delta(q, x, A)$, а тогда $(q, x, A) \underset{M}{\vdash} (q, \varepsilon, \alpha)$.

Индукционная гипотеза. Предположим, что для всех $l \le n$ ($n \ge 1$), если $A \overset{l}{\underset{G}{\Rightarrow}} x\alpha$, то $(q, x, A) \overset{*}{\underset{M}{\vdash}} (q, \varepsilon, \alpha)$.

Индукционный переход. Докажем это утверждение для $l = n + 1$. Пусть $A \overset{n}{\underset{G}{\Rightarrow}} yB\gamma \underset{G}{\Rightarrow} ya\beta\gamma = x\alpha$ — левосторонний вывод длиной $n + 1$. В нем $x = ya$, $\alpha = \beta\gamma$ и $B \to a\beta \in P$. По построению автомата M существует $(q, \beta) \in \delta(q, a, B)$. В соответствии с индукционной гипотезой и с учетом этого последнего обстоятельства можем написать: $(q, x, A) = (q, ya, A) \overset{*}{\underset{M}{\vdash}} (q, a, B\gamma) \underset{M}{\vdash} (q, \varepsilon, \beta\gamma) = (q, \varepsilon, \alpha)$.

II. Теперь индукцией по числу l движений автомата M покажем, что если $(q, x, A) \overset{l}{\underset{M}{\vdash}} (q, \varepsilon, \alpha)$, где $x \in V_T^*$, $\alpha \in V_N^*$, то $A \overset{*}{\underset{G}{\Rightarrow}} x\alpha$.

База. Пусть $l = 1$. Имеем $(q, x, A) \underset{M}{\vdash} (q, \varepsilon, \alpha)$, где $x \in V_T^*$, $\alpha \in V_N^*$. Поскольку pda M не совершает ε-движений, то это движение имеет место благодаря тому, что $(x, \alpha) \in \delta(q, x, A)$, где $x \in V_T$. Но это обусловлено тем, что существует правило $A \to x\alpha \in P$, с помощью которого получаем требуемый вывод $A \underset{G}{\Rightarrow} x\alpha$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \le n$ ($n \ge 1$).

Индукционный переход. Докажем это утверждение для $l = n + 1$. Пусть $(q, x, A) = (q, ya, A) \overset{n}{\underset{M}{\vdash}} (q, a, B\gamma) \underset{M}{\vdash} (q, \varepsilon, \beta\gamma) = (q, \varepsilon, \alpha)$, где $x = ya$, $\alpha = \beta\gamma$. Последнее из этих движений подразумевает, что $(q, \beta) \in \delta(q, a, B)$, где $a \in V_T$, а это обусловлено существованием правила $B \to a\beta \in P$. Учитывая следствие индукционной гипотезы из $(q, ya, A) \overset{n}{\underset{M}{\vdash}} (q, a, B\gamma)$ и упомянутое правило, мы можем написать:

$$A \overset{*}{\underset{G}{\Rightarrow}} yB\gamma \underset{G}{\Rightarrow} ya\beta z\gamma = x\alpha.$$

Из рассуждений, проведенных в пп. I и II, при $A = S$ и $\alpha = \varepsilon$ получаем утверждение теоремы. Что и требовалось доказать.

Теорема 5.3. *Если M — недетерминированный магазинный автомат, и $L = N(M)$, то L — контекстно-свободный язык.*

Доказательство. Пусть $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \varnothing)$ — недетерминированный pda, такой, что $L = N(M)$.

Построим cfg G, левосторонние выводы в которой моделируют движения pda M. В частности, нетерминалы, которые появляются в сентенциальных формах на любом шаге левостороннего вывода в грамматике G, соответствуют символам в магазине автомата M в момент, когда он уже просмотрел такую же часть входной цепочки, какую грамматика уже породила.

Положим $G = (V_N, V_T, P, S)$, где $V_N = \{S\} \cup \{[qAp] \mid q, p \in Q; A \in \Gamma\}$; $V_T = \Sigma$;

$P = \{(1)\, S \to [q_0 Z_0 q] \mid q \in Q\} \cup$

$\{(2)\, [qAp] \to a[q_1 B_1 q_2][q_2 B_2 q_3]...[q_m B_m q_{m+1}] \mid q, qz_i \in Q\ (i = 1, 2, ..., m+1);$

$p = q_{m+1};\ a \in \Sigma \cup \{\varepsilon\};\ A, B_j \in \Gamma\ (j = 1, 2, ..., m);\ (q_1, B_1 B_2 ... B_m) \in \delta(q, a, A)\}$.

Отметим, что если $(q_1, \varepsilon) \in \delta(q, a, A)$, то $m = 0$, $q_1 = p$ и правило вида 2 имеет вид $[qAp] \to a$.

Чтобы показать, что $L(G) = N(M)$, мы сначала докажем вспомогательное утверждение: $[qAp] \overset{*}{\underset{G}{\Rightarrow}} x$ тогда и только тогда, когда $(q, x, A) \overset{*}{\underset{M}{\vdash}} (p, \varepsilon, \varepsilon)$.

I. Индукцией по числу l движений pda M покажем, что если $(q, x, A) \overset{*}{\underset{M}{\vdash}} (p, \varepsilon, \varepsilon)$, то $[qAp] \overset{*}{\underset{G}{\Rightarrow}} x$.

База. Пусть $l = 1$. Имеем $(q, x, A) \underset{M}{\vdash} (p, \varepsilon, \varepsilon)$. Это движение обусловлено тем, что $(p, \varepsilon) \in \delta(q, x, A)$, где $x \in \Sigma \cup \{\varepsilon\}$. Соответственно по способу построения грамматики G существует правило $[qAp] \to x$, при помощи которого $[qAp] \underset{G}{\Rightarrow} x$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех значений $l \le n$ $(n \ge 1)$.

Индукционный переход. Докажем, что утверждение выполняется для $l = n + 1$. Пусть $(q, x, A) = (q, ax_1x_2... x_n, A) \underset{M}{\vdash} (q_1, x_1x_2... x_n, B_1B_2...B_m) \overset{n}{\underset{M}{\vdash}} (p, \varepsilon, \varepsilon)$. Здесь $x = ax_1x_2 ... x_n$, $a \in \Sigma \cup \{\varepsilon\}$, $x_i \in \Sigma^*$, причем $(q_i, x_i, B_i) \overset{l_i}{\underset{M}{\vdash}} (q_{i+1}, \varepsilon, \varepsilon)$, $i = 1, 2, ..., m$, $q_{m+1} = p$.

Поясним, что в этой последовательности переходов явно показано первое движение, которое может использовать входной символ a, с которого начинается входная цепочка x, либо оно может быть ε-движением. Каждая из подцепочек x_i есть та часть цепочки x, которую автомат просматривает с момента, когда он оказывается в состоянии q_i с символом B_i на вершине магазина, до того момента, когда вершина магазина впервые опустится на один символ ниже этой позиции B_i.

Очевидно, что первое движение обусловлено тем, что $(q_1, B_1B_2 ... B_m) \in \delta(q, a, A)$, и, как следствие этого, существует правило грамматики вида

$$[qAp] \to a[q_1 B_1 q_2][q_2 B_2 q_3]...[q_m B_m q_{m+1}]$$

с нетерминалами, связанными с состояниями и магазинными символами, участвующими в указанных конфигурациях. С другой стороны, применение индукционной гипотезы к переходу $(q_i, x_i, B_i) \overset{l_i}{\underset{M}{\vdash}} (q_{i+1}, \varepsilon, \varepsilon)$, где $l_i \le n$, дает основание заключить, что $[q_i B_i q_{i+1}] \overset{*}{\underset{G}{\Rightarrow}} x_i$, $i = 1, 2, \ldots, m$, $q_{m+1} = p$.

С помощью упомянутых правила и частичных выводов можно выстроить требуемый вывод:

$[qAp] \underset{G}{\Rightarrow} a[q_1B_1q_2][q_2B_2q_3]\ldots[q_mB_mq_{m+1}] \overset{*}{\underset{G}{\Rightarrow}} ax_1[q_2B_2q_3]\ldots[q_mB_mq_{m+1}] \overset{*}{\underset{G}{\Rightarrow}} \ldots$
$\overset{*}{\underset{G}{\Rightarrow}} ax_1x_2\ldots x_n = x.$

II. Индукцией по длине вывода l покажем, что если $[qAp] \overset{l}{\underset{G}{\Rightarrow}} x$, то $(q, x, A) \overset{*}{\underset{M}{\vdash}} (p, \varepsilon, \varepsilon)$.

База. Пусть $l = 1$. Имеем $[qAp] \underset{G}{\Rightarrow} x$. Очевидно, что на этом единственном шаге применено правило $[qAp] \to x$. По способу построения cfg G $x \in \Sigma \cup \{\varepsilon\}$. Такое правило существует, если только $(p, \varepsilon) \in \delta(q, x, A)$. А тогда $(q,x,A) \underset{M}{\vdash} (p, \varepsilon, \varepsilon)$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех значений $l \le n$ $(n \ge 1)$.

Индукционный переход. Докажем, что утверждение выполняется для $l = n + 1$. Пусть $[qAp] \underset{G}{\Rightarrow} a[q_1B_1q_2][q_2B_2q_3]\ldots[q_mB_mq_{m+1}] \overset{n}{\underset{G}{\Rightarrow}} ax_1x_2\ldots x_n = x$. Очевидно, что существует правило $[qAp] \to a[q_1B_1q_2][q_2B_2q_3]\ldots[q_mB_mq_{m+1}] \in P$, использованное на первом шаге вывода, в котором $q_{m+1} = p$, и частичные выводы $[q_iB_iq_{i+1}] \overset{l_i}{\underset{G}{\Rightarrow}} x_i$, где $l_i \le n$, $i = 1, 2, \ldots, m$. Из существования такого правила следует, что $(q_1, B_1B_2\ldots B_m) \in \delta(q, a, A)$. Из частичных выводов в соответствии с индукционной гипотезой вытекает, что $(q_i, x_i, B_i) \overset{l_i}{\underset{M}{\vdash}} (q_{i+1}, \varepsilon, \varepsilon)$, $i = 1, 2, \ldots, m$, $q_{m+1} = p$. Следовательно, существует последовательность конфигураций:

$$(q, x, A) = (q, ax_1x_2\ldots x_n, A) \underset{M}{\vdash} (q_1, x_1x_2\ldots x_n, B_1B_2\ldots B_m) \overset{*}{\underset{M}{\vdash}}$$
$$\overset{*}{\underset{M}{\vdash}} (q_2, x_2\ldots x_n, B_2\ldots B_m) \overset{*}{\underset{M}{\vdash}} \ldots \overset{*}{\underset{M}{\vdash}} (p, \varepsilon, \varepsilon).$$

Итак, из рассуждений, проведенных в пп. I и II, следует справедливость вспомогательного утверждения: $[qAp] \overset{*}{\underset{G}{\Rightarrow}} x$ тогда и только тогда, когда $(q, x, A) \overset{*}{\underset{M}{\vdash}} (p, \varepsilon, \varepsilon)$. В частности, при $q = q_0$ и $A = Z_0$ $[q_0Z_0p] \overset{*}{\underset{G}{\Rightarrow}} x$ тогда и только тогда, когда $(q_0, x, Z_0) \overset{*}{\underset{M}{\vdash}} (p, \varepsilon, \varepsilon)$.

Остается вспомнить, что существуют правила вида $S \to [q_0Z_0q]$ для любых $q \in Q$, в частности для $q = p$; тогда, пристраивая новое начало к выводу, получаем справедливость следующего утверждения: $S \underset{G}{\Rightarrow} [q_0Z_0p] \overset{*}{\underset{G}{\Rightarrow}} x$ тогда и только тогда, когда $(q_0, x, Z_0) \overset{*}{\underset{M}{\vdash}} (p, \varepsilon, \varepsilon)$, или, что то же самое, $L(G) = N(M)$. Что и требовалось доказать.

Теоремы 5.1, 5.2 и 5.3 можно подытожить: следующие три утверждения являются эквивалентными:

1) L — cfl;
2) $L = N(M_1)$ для некоторого pda M_1;
3) $L = T(M_2)$ для некоторого pda M_2.

Пример 5.3. Пусть pda $M = (\{q_0, q_1\}, \{0, 1\}, \{X, Z_0\}, \delta, q_0, Z_0, \varnothing)$,
где $\delta = \{$ (1) $\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$, (4) $\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$,
(2) $\delta(q_0, 0, X) = \{(q_0, XX)\}$, (5) $\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$,
(3) $\delta(q_0, 1, X) = \{(q_1, \varepsilon)\}$, (6) $\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}\}$.

Нетрудно сообразить, что $N(M) = \{0^n 1^m \mid n \geq m\}$.

Построим cfg $G = (V_N, V_T, P, S)$, порождающую язык $N(M)$, положив
$V_N = \{S, [q_0Z_0q_0], [q_0Z_0q_1], [q_1Z_0q_0], [q_1Z_0q_1], [q_0Xq_0], [q_0Xq_1], [q_1Xq_0], [q_1Xq_1]\}$,
$V_T = \{0, 1\}$.

Для экономии времени при построении правил грамматики следует учесть, что некоторые нетерминалы не участвуют ни в каких выводах, начинающихся с символа S. Поэтому мы должны начать с построения правил для S, затем построить правила для тех нетерминалов, которые встречаются в правых частях уже построенных правил.

Правилами для S являются: $S \to [q_0Z_0q_0]$ и $S \to [q_0Z_0q_1]$.

Затем, исходя из правила 1 в определении δ, мы добавим правила для нетерминала $[q_0Z_0q_0]$:

$$[q_0Z_0q_0] \to 0[q_0Xq_0][q_0Z_0q_0],$$
$$[q_0Z_0q_0] \to 0[q_0Xq_1][q_1Z_0q_0].$$

Так как в правых частях построенных правил появился нетерминал $[q_0Z_0q_1]$, мы должны построить правила для него, опять исходя из того же правила 1:

$$[q_0Z_0q_1] \to 0[q_0Xq_0][q_0Z_0q_1],$$
$$[q_0Z_0q_1] \to 0[q_0Xq_1][q_1Z_0q_1].$$

Аналогично, исходя из правила 2 в определении δ, получаем следующие правила грамматики:

$$[q_0Xq_0] \to 0[q_0Xq_0][q_0Xq_0],$$
$$[q_0Xq_0] \to 0[q_0Xq_1][q_1Xq_0],$$
$$[q_0Xq_1] \to 0[q_0Xq_0][q_0Xq_1],$$
$$[q_0Xq_1] \to 0[q_0Xq_1][q_1Xq_1].$$

Исходя из правил 3–6 в определении δ, получаем следующие правила грамматики:

$$[q_0Xq_1] \to 1,\ [q_1Xq_1] \to 1,\ [q_1Xq_1] \to \varepsilon,\ [q_1Z_0q_1] \to \varepsilon.$$

Заметим, что нет никаких правил для нетерминалов $[q_1Z_0q_0]$ и $[q_1Xq_0]$. Соответственно правила, в правых частях которых встречаются эти два нетерминала, бесполезны и могут быть отброшены. Кроме того, легко заметить, что из нетерминалов $[q_0Z_0q_0]$ и $[q_0Xq_0]$ не выводится ни одной терминальной цепочки. Поэтому правила с их участием могут быть отброшены.

Окончательно получаем следующее множество правил приведенной грамматики:

(1) $S \to [q_0Z_0q_1]$, (4) $[q_0Xq_1] \to 1$,
(2) $[q_0Z_0q_1] \to 0[q_0Xq_1][q_1Z_0q_1]$, (5) $[q_1Xq_1] \to 1$,
(3) $[q_0Xq_1] \to 0[q_0Xq_1][q_1Xq_1]$, (6) $[q_1Xq_1] \to \varepsilon$, (7) $[q_1Z_0q_1] \to \varepsilon$.

Глава 6

МАШИНЫ ТЬЮРИНГА

§ 6.1. Неформальное и формальное описания

В этой главе мы рассмотрим еще один тип распознающих устройств — *машины Тьюринга*. Это абстрактное устройство было предложено в качестве математической модели для описания процедур. Так как наше интуитивное понятие процедуры как конечной последовательности инструкций, которые могут выполняться механически, не является математически точным, мы не можем доказать формально, что оно эквивалентно точному понятию машины Тьюринга. Однако из определения этого устройства будет очевидно, что любое вычисление, которое может быть описано посредством машины Тьюринга, может быть выполнено механически. Также может быть показано, что любое вычисление, которое может быть выполнено на современной вычислительной машине, может быть описано посредством машины Тьюринга. Таким образом, если кто-нибудь когда-нибудь нашел бы процедуру, которая соответствует интуитивным понятиям, но не поддается описанию посредством машины Тьюринга, то она была бы необычной природы, так как не могла бы быть запрограммирована для любой существующей вычислительной машины. Было предложено много других формализаций процедуры, и было показано, что все они эквивалентны формализации машины Тьюринга. Это поддерживает нашу уверенность в том, что машина Тьюринга имеет достаточную общность, чтобы охватить интуитивное понятие процедуры.

А. Чёрчем была высказана гипотеза, что любой процесс, который естественным образом мог бы быть назван процедурой, реализуем машиной Тьюринга. Впоследствии вычислимость при помощи машины Тьюринга стала признанным определением процедуры. Мы примем гипотезу Чёрча и просто подставим формальное определение машины Тьюринга вместо интуитивного понятия процедуры.

В литературе определение машины Тьюринга давалось разными способами. Мы начнем с обсуждения основной модели (рис. 6.1).

Основная модель имеет *конечное управление*, *ленту*, которая разделена на *ячейки*, и *головку ленты*, которая сканирует одну ячейку ленты в один прием. Лента имеет крайнюю левую ячейку, но простирается в бесконечность в правую сторону. Каждая ячейка может содержать ровно один из конечного числа *символов ленты*. Первоначально n крайних левых ячеек для некоторого конечного n содержит *входную цепочку*, строку символов ленты, называемых *входными символами*. Остальные ячейки до бесконечности содержат *пробел* — специальный символ ленты, который не является входным символом.

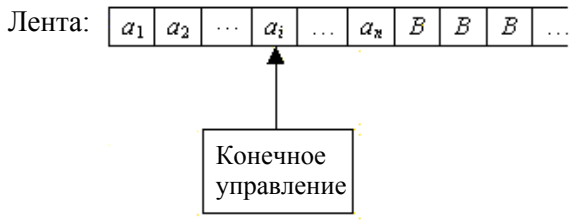

Рис. 6.1.

В один *такт*, зависящий от символа, сканируемого головкой ленты, и состояния конечного управления, машина Тьюринга

1) изменяет состояние;

2) печатает символ ленты, не являющийся пробелом, в сканируемой ячейке ленты, замещая то, что было там записано;

3) сдвигает свою головку влево или вправо на одну ячейку.

***Определение* 6.1.** *Машина Тьюринга* (Tm) является формальной системой: $T = (Q, \Sigma, \Gamma, \delta, q_0, F)$, где Q — конечное *множество состояний*; Γ — конечное *множество допустимых символов ленты*, один из них, обычно обозначаемый буквой B, есть *пробел*; $\Sigma \subseteq \Gamma \setminus \{B\}$ — *множество входных символов*; $\delta : Q \times \Gamma \to Q \times (\Gamma \setminus \{B\}) \times \{L, R\}$ — *функция следующего такта* (*движения*), для некоторых аргументов может быть не определена[8]; $q_0 \in Q$ — *начальное состояние*; $F \subseteq Q$ — *множество конечных состояний*.

***Определение* 6.2.** *Конфигурацией* машины Тьюринга назовем тройку (q, α, i), где $q \in Q$ — текущее состояние машины Тьюринга; $\alpha \in (\Gamma \setminus \{B\})^*$ — строка, являющаяся непустой частью ленты; i — целое, определяющее позицию головки ленты, отсчитываемую от левого конца ленты.

Заметим, что если головка ленты покидает ячейку, она должна напечатать непустой символ в этой ячейке, так что лента всегда содержит непустой блок символов (α — этот блок) с бесконечным числом пробелов справа от него.

Определим теперь один такт (движение) машины Тьюринга T. Пусть $(q, A_1A_2 \ldots A_n, i)$ — некоторая ее конфигурация, где $1 \le i \le n + 1$.

Случай 1. Если $1 \le i \le n$ и $\delta(q, A_i) = (p, A, R)$, то

$$(q, A_1A_2 \ldots A_n, i) \vdash_T (p, A_1A_2 \ldots A_{i-1}AA_{i+1} \ldots A_n, i + 1),$$

т.е. T печатает символ A в i-й позиции и двигается вправо.

Случай 2. Если $2 \le i \le n$ и $\delta(q, A_i) = (p, A, L)$, то

$$(q, A_1A_2 \ldots A_n, i) \vdash_T (p, A_1A_2 \ldots A_{i-1}A\, A_{i+1} \ldots A_n, i - 1),$$

т.е. T печатает символ A в i-й позиции и двигается влево, не сходя с левого конца ленты.

---

[8] Мы не позволили Tm печатать пробел ради простоты определения конфигураций. Однако Tm могла бы иметь другой символ, который трактуется точно так же, как пробел, за исключением того, что Tm разрешается печатать этот символ псевдопробела. Разумеется, никакой дополнительной мощности не появляется за счет введения такого символа. В неформальном обсуждении мы часто допускаем печать пробела, зная, что вместо него можно использовать другой, но эквивалентный ему символ.

Случай 3. Если $i = n + 1$, то головка сканирует пробел B.

а) Если при этом $\delta(q, B) = (p, A, R)$, то

$$(q, A_1A_2 \ldots A_n, n + 1) \vdash_T (p, A_1A_2 \ldots A_nA, n + 2).$$

б) Если при этом $\delta(q, B) = (p, A, L)$, то

$$(q, A_1A_2 \ldots A_n, n + 1) \vdash_T (p, A_1A_2 \ldots A_nA, n).$$

Таким образом, мы ввели отношение непосредственного следования одной конфигурации за другой. Очевидным образом можно определить степень, транзитивное и рефлексивно-транзитивное замыкания этого отношения. Будем обозначать их традиционно через $\vdash_T^n$, $\vdash_T^+$ и $\vdash_T^*$ соответственно. Если две конфигурации связаны знаком $\vdash_T^n$, то мы будем говорить, что вторая получается из первой за n движений. Соответственно запись $\vdash_T^+$ обозначает положительное число движений, а $\vdash_T^*$ — любое число движений, включая нуль.

***Определение* 6.3.** Пусть $T = (Q, \Sigma, \Gamma, \delta, q_0, F)$ — машина Тьюринга. *Язык*, *принимаемый* машиной T есть $L = \{w \mid w \in \Sigma^*$ и $(q_0, w, 1) \vdash_T^* (q, \alpha, i)$ для некоторых $q \in F$, $\alpha \in \Gamma^*$ и $i > 0\}$.

Мы предполагаем, что данная Tm, распознающая язык L, *останавливается*, т.е. не имеет никакого следующего движения, всякий раз, как входная цепочка принимается. Но для цепочек, которые не принимаются, Tm может не остановиться.

Пример 6.1. Построим Tm, распознающую cfl $L = \{0^n1^n \mid n \geq 1\}$. Положим $T = (Q, \Sigma, \Gamma, \delta, q_0, F)$, где $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$; $\Sigma = \{0,1\}$; $\Gamma = \{0, 1, B, X, Y\}$; $F = \{q_5\}$. Функцию δ определим следующим образом:

1. $\delta(q_0, 0) = (q_1, X, R)$.

В состоянии q_0 символ 0 заменяется на X и машина сдвигается вправо в состояние q_1 в поисках 1.

2. а) $\delta(q_1, 0) = (q_1, 0, R)$;
б) $\delta(q_1, Y) = (q_1, Y, R)$;
в) $\delta(q_1, 1) = (q_2, Y, L)$.

Оставаясь в состоянии q_1, машина продвигается вправо сквозь все нули (п. 2а) и блок Y (п. 2б). Наткнувшись на 1, заменяет ее на Y и переходит в состояние q_2, начав движение влево (п. 2в).

3. а) $\delta(q_2, Y) = (q_2, Y, L)$;
б) $\delta(q_2, X) = (q_3, X, R)$;
в) $\delta(q_2, 0) = (q_4, 0, L)$.

Оставаясь в состоянии q_2, машина продвигается влево сквозь блок Y (п. 3а). Если машина встречает X, все еще оставаясь в состоянии q_2, то больше нет нулей, которые следовало бы заменять на X, и машина переходит в состояние q_3, начиная движение вправо, чтобы убедиться, что не осталось единиц (п. 3б). Если же 0 встретился, машина переходит в состояние q_4, чтобы продолжить движение в поисках крайнего левого 0 (п. 3в).

4. а) $\delta(q_4, 0) = (q_4, 0, L)$

б) $\delta(q_4, X) = (q_0, X, R)$.

Машина движется сквозь нули (п. 4а). Если встретился X, то машина прошла самый левый нуль. Она должна, сдвинувшись вправо, превратить этот 0 в X (п. 4б). Происходит переход в состояние q_0, и процесс, только что описанный в п. 1–4, продолжается.

5. а) $\delta(q_3, Y) = (q_3, Y, R)$

б) $\delta(q_3, B) = (q_5, Y, R)$.

Машина входит в состояние q_3, когда ни одного 0 не остается (см. п. 3б). Машина должна продвигаться вправо (п. 5а) сквозь блок Y. Если встречается пробел раньше, чем 1, то ни одной 1 не осталось (п. 5б). В этой ситуации машина переходит в конечное состояние q_5 и останавливается, сигнализируя тем самым прием входной цепочки.

6. Во всех случаях, кроме 1–5, функция δ не определена.

Рассмотрим действия машины Тьюринга на входной цепочке 000111.

Табл. 6.1

Шаг	Конфигурация	Шаг	Конфигурация	Шаг	Конфигурация
1	$0\ 0\ 1\ 1\ 1$ q_0	10	$X\ X\ 0\ Y\ 1\ 1$ q_1	19	$X\ X\ X\ Y\ Y\ Y$ q_2
2	$X\ 0\ 0\ 1\ 1\ 1$ q_1	11	$X\ X\ 0\ Y\ 1\ 1$ q_1	20	$X\ X\ X\ Y\ Y\ Y$ q_2
3	$X\ 0\ 0\ 1\ 1\ 1$ q_1	12	$X\ X\ 0\ Y\ Y\ 1$ q_2	21	$X\ X\ X\ Y\ Y\ Y$ q_2
4	$X\ 0\ 0\ 1\ 1\ 1$ q_1	13	$X\ X\ 0\ Y\ Y\ 1$ q_2	22	$X\ X\ X\ Y\ Y\ Y$ q_3
5	$X\ 0\ 0\ Y\ 1\ 1$ q_2	14	$X\ X\ 0\ Y\ Y\ 1$ q_4	23	$X\ X\ X\ Y\ Y\ Y$ q_3
6	$X\ 0\ 0\ Y\ 1\ 1$ q_4	15	$X\ X\ 0\ Y\ Y\ 1$ q_0	24	$X\ X\ X\ Y\ Y\ Y$ q_3
7	$X\ 0\ 0\ Y\ 1\ 1$ q_4	16	$X\ X\ X\ Y\ Y\ 1$ q_1	25	$X\ X\ X\ Y\ Y\ Y$ q_3
8	$X\ 0\ 0\ Y\ 1\ 1$ q_0	17	$X\ X\ X\ Y\ Y\ 1$ q_1	26	$X\ X\ X\ Y\ Y\ Y\ Y$ q_5
9	$X\ X\ 0\ Y\ 1\ 1$ q_1	18	$X\ X\ X\ Y\ Y\ 1$ q_1		

В табл. 6.1 приведены конфигурации в виде цепочек символов ленты с маркером состояния под сканируемым символом (в конфигурациях 25 и 26 маркер состояния находится под символом пробела).

§ 6.2. Методы построения машин Тьюринга

Машина Тьюринга может “программироваться” во многом так же, как программируются вычислительные машины. Роль программы играет функция δ. В этом параграфе мы представим коллекцию приемов программирования машины Тьюринга, которые помогут лучше узнать ее возможности.

6.2.1. Конечное управление в роли запоминающего устройства. Конечное управление может использоваться для запоминания конечного количества информации. Именно: состояние записывается как пара элементов, причем один осуществляет управление, а другой запоминает символ. Подчеркнем, что этот прием используется только концептуально. Никакой модификации основной модели машины Тьюринга не подразумевается.

Пример 6.2. Пусть $T = (Q, \{0, 1\}, \{0, 1, B\}, \delta, [q_0, B], F)$, где $Q = \{[q_0, 0], [q_0, 1], [q_0, B], [q_1, 0], [q_1, 1], [q_1, B]\}$, $F = \{[q_1, B]\}$, т.е. здесь Q записано как $\{q_0, q_1\} \times \{0, 1, B\}$.

Запрограммируем распознавание языка, состоящего из цепочек, в которых первый символ повторно не встречается в той же самой цепочке.

Отметим, что такой язык является регулярным.

Построим функцию δ следующим образом:

1. а) $\delta([q_0, B], 0) = ([q_1, 0], 0, R)$;
б) $\delta([q_0, B], 1) = ([q_1, 1], 1, R)$.

Машина запоминает сканируемый символ во второй компоненте обозначения состояния и сдвигается вправо. Первой компонентой становится q_1.

2. а) $\delta([q_1, 0], 1) = ([q_1, 0], 1, R)$;
б) $\delta([q_1, 1], 0) = ([q_1, 1], 0, R)$.

Если машина помнит 0 и видит 1 или, наоборот, помнит 1 и видит 0, то она продолжает движение вправо.

3. а) $\delta([q_1, 0], B) = ([q_1, B], 0, L)$;
б) $\delta([q_1, 1], B) = ([q_1, B], 0, L)$.

Машина входит в конечное состояние $[q_1, B]$, если она встречает символ пробела раньше, чем достигает второй копии самого левого символа. Если же машина достигает пробела в состоянии $[q_1, 0]$ или $[q_1, 1]$, то она принимает входную цепочку. Для состояния $[q_1, 0]$ и символа 0 или для состояния $[q_1, 1]$ и символа 1 функция δ не определена, так что, если машина когда-нибудь видит запомненный символ, она останавливается, не принимая.

В общем случае можно допустить произвольное фиксированное число компонент, причем все, кроме одной, предназначены для запоминания информации.

6.2.2. Многодорожечные ленты. Мы можем подразумевать, что лента машины Тьюринга разделена на k дорожек для любого конечного k. На рис. 6.2

представлено такое устройство для $k = 3$. Прием состоит в том, что символы на ленте рассматриваются как наборы из k элементов, по одному на каждой дорожке.

Пример 6.3. Представим, что лента, показанная на рис. 6.2, является лентой машины Тьюринга, которая принимает двоичный код целых чисел, значение которых больше 2. Этот код записан на верхней дорожке и ограничен слева и справа символами ¢ и $.

Рис. 6.2.

Задача машины — определить, является ли закодированное число простым. Мы не будем строить фактическую программу этой машины, а только обсудим способ кодирования информации на 3-дорожечной ленте и дадим словесное описание ее работы.

В начальный момент на верхней дорожке, как уже сказано, находится двоичный код тестируемого числа с упомянутыми ограничителями слева и справа. Две другие дорожки пусты. Таким образом, входными символами являются тройки $[¢, B, B]$, $[0, B, B]$, $[1, B, B]$ и $[\$, B, B]$. Они отождествляются соответственно с ¢, 0, 1 и $. Символ пробела можно представить как $[B, B, B]$.

Чтобы проверить, находится ли на ее ленте код простого числа, машина, во-первых, записывает двоичный код числа 2 на второй дорожке и копирует первую дорожку на третью. Затем вторая дорожка вычитается из третьей столько раз, сколько можно, фактически оставляя остаток на третьей дорожке. Если остаток есть нуль, то число на первой дорожке не простое. Если остаток не нуль, то число на второй дорожке увеличивается на 1. Если теперь вторая дорожка равна первой, число на первой дорожке простое, поскольку оно не делится без остатка на любое число между единицей и самим собой. Если вторая дорожка меньше первой, вся операция повторяется для нового числа на второй дорожке и восстановленной копии первой дорожки на третьей.

На рис. 6.2 машина определяет, является ли число 47 простым. В показанный момент она делит 47 на 5 и уже дважды вычитание 5 было выполнено, так что на третьей дорожке содержится промежуточный результат: 37.

6.2.3. Отметка символов является полезным приемом, чтобы увидеть, как машина Тьюринга распознает языки, предложения которых состоят из повторяющихся строк, например $\{ww \mid w \in \Sigma^*\}$, $\{wcy \mid w, y \in \Sigma^*, w \neq y\}$ или $\{ww^R \mid w \in \Sigma^*\}$.

Этот прием полезен также тогда, когда необходимо сравнивать длины подцепочек, как, например, в языках $\{a^i b^i \mid i \geq 1\}$ или $\{a^i b^j c^k \mid i \neq j \text{ или } j \neq k\}$. Прием состоит в том, что вводится дополнительная дорожка на ленте машины Тью-

ринга, которая содержит пробел или символ √. Последний используется для отметки того, что символ, расположенный под ним, уже рассмотрен машиной в одном из предыдущих сравнений.

Пример 6.4. Построим машину Тьюринга, которая распознает язык $L=\{wcw \mid w\in \{a, b\}^+\}$. Положим $T = (Q, \Sigma, \Gamma, \delta, q_0, F)$, где $Q = \{[q, d] \mid q\in \{q_1, q_2,..., q_9\}, d\in \{a, b, B\}\}$; вторая компонента состояния используется для того, чтобы запомнить входной символ; $\Sigma = \{[B, d] \mid d\in \{a, b, c\}\}$; $\Gamma = \{[X, d] \mid X\in \{B, √\}, d\in \{a, b, c, B\}\}$, $q_0 = [q_1, B]$, $F = \{[q_9, B]\}$. Символ пробела представляется как $[B, B]$, символ a идентифицируется с $[B, a]$, символ b идентифицируется с $[B, b]$ и символ c идентифицируется с $[B, c]$.

Определим функцию δ следующим образом:

1. $\delta([q_1, B], [B, d]) = ([q_2, d], [√, d], R)$ для $d\in \{a, b\}$.

Машина отмечает сканируемый символ на ленте, запоминает его в конечном управлении и начинает движение вправо, переходя в новое состояние с первой компонентой q_2.

2. $\delta([q_2, d], [B, e]) = ([q_2, d], [B, e], R)$ для $d,e\in \{a, b\}$.

Машина продолжает двигаться вправо по непроверенным символам в поисках символа c.

3. $\delta([q_2, d], [B, c]) = ([q_3, d], [B, c], R)$ для $d\in \{a, b\}$.

По нахождению символа c машина входит в новое состояние с первой компонентой q_3.

4. $\delta([q_3, d], [√, e]) = ([q_3, d], [√, e], R)$ для $d,e\in \{a, b\}$.

Машина движется вправо по проверенным символам.

5. $\delta([q_3, d], [B, d]) = ([q_4, B], [√, d], L)$ для $d\in \{a, b\}$.

Если машина “видит” неотмеченный символ, равный запомненному в конечном управлении, то отмечает его и начинает движение влево, переходя в новое состояние с первой компонентой q_4 и “забывая” символ, который она только что отметила. В противном случае дальнейшее движение не определено: машина останавливается, не принимая.

6. $\delta([q_4, B], [√, d]) = ([q_4, B], [√, d], L)$ для $d\in \{a, b\}$.

Машина движется влево по проверенным символам.

7. $\delta([q_4, B], [B, c]) = ([q_5, B], [B, c], L)$.

Машина встречает символ c, фиксирует этот факт переходом в новое состояние $[q_5, B]$ и продолжает движение влево.

8. $\delta([q_5, B], [B, d]) = ([q_6, B], [B, d], L)$ для $d\in \{a, b\}$.

Если символ непосредственно слева от c не отмечен, то машина фиксирует этот факт переходом в новое состояние $[q_6, B]$ и, продолжая двигаться влево, начинает поиск самого левого из непомеченных символов.

9. $\delta([q_6, B], [B, d]) = ([q_6, B], [B, d], L)$ для $d\in \{a, b\}$.

Машина продолжает движение влево сквозь непомеченные символы.

10. $\delta([q_6, B], [\surd, d]) = ([q_1, B], [\surd, d], R)$ для $d \in \{a, b\}$.

Машина встречает помеченный символ и сдвигается вправо, чтобы взять следующий символ для сравнения. Первой компонентой состояния снова становится q_1. С этого момента цикл запоминания очередного непроверенного символа и его поиска на правой половине входной цепочки повторяется.

11. $\delta([q_5, B], [\surd, d]) = ([q_7, B], [\surd, d], R)$ для $d \in \{a, b\}$.

В состоянии $[q_5, B]$ машина окажется сразу после прохождения символа c при попятном движении (см. п. 7). Если символ непосредственно слева от c отмечен, то это значит, что все символы на левой половине цепочки проверены. Остается проверить, все ли символы и на правой половине тоже отмечены. Если это так, то цепочка слева от c равна цепочке справа от c, и машина останавливается, принимая входную цепочку. Иначе — машина останавливается, не принимая эту цепочку.

12. $\delta([q_7, B], [B, c]) = ([q_8, B], [B, c], R)$.

Машина сдвигается вправо от c, фиксируя переходом в новое состояние $[q_8, B]$ режим проверки отметки всех символов на правой половине цепочки.

13. $\delta([q_8, B], [\surd, d]) = ([q_8, B], [\surd, d], R)$ для $d \in \{a, b\}$.

Машина двигается вправо через отмеченные символы на правой половине цепочки.

14. $\delta([q_8, B], [B, B]) = ([q_9, B], [\surd, B], L)$.

Если машина находит пробел $[B, B]$, она останавливается и принимает входную цепочку. Если машина находит неотмеченный символ, когда первая компонента ее состояния — q_8, она останавливается, не принимая входной цепочки.

6.2.4. Сдвиг символов ленты. Машина Тьюринга может освобождать место на своей ленте, сдвигая все непустые символы на конечное число ячеек вправо. Она может затем вернуться к освобожденным ячейкам и печатать символы по своему выбору. Если пространство доступно, машина может сдвигать блоки символов влево подобным же образом.

Пример 6.5. Построим часть машины Тьюринга $T = (Q, \Sigma, \Gamma, \delta, q_0, F)$, которая время от времени имеет необходимость сдвигать символы на две ячейки вправо. Пусть Q содержит состояния вида $[q, A_1, A_2]$ для $q \in \{q_1, q_2\}$ и $A_1, A_2 \in \Gamma$. Пусть B — пробел, а X — специальный символ, не используемый машиной нигде, кроме как в процессе сдвига. Мы предполагаем, что машина начинает процесс сдвига в состоянии $[q_1, B, B]$. Относящаяся к этому часть функции δ определяется следующим образом:

1. $\delta([q_1, B, B], A_1) = ([q_1, B, A_1], X, R)$ для $A_1 \in \Gamma \setminus \{B, X\}$.

Машина запоминает первый прочитанный символ в третьей компоненте ее состояния, печатает на его месте символ X и начинает движение вправо.

2. $\delta([q_1, B, A_1], A_2) = ([q_1, A_1, A_2], X, R)$ для $A_1, A_2 \in \Gamma \setminus \{B, X\}$.

Машина запоминает прочитанный символ в третьей компоненте ее состояния, сдвигая перед этим третью компоненту на место второй, печатает на его месте символ X и двигается вправо.

3. $\delta([q_1, A_1, A_2], A_3) = ([q_1, A_2, A_3], A_1, R)$ для $A_1, A_2, A_3 \in \Gamma \setminus \{B, X\}$.

Машина запоминает прочитанный символ в третьей компоненте ее состояния, сдвигая перед этим третью компоненту на место второй, печатает на его месте символ A_1 из прежней второй компоненты состояния и двигается вправо. Очевидно, что символ A_1 будет напечатан двумя ячейками правее той позиции, в которой он находился прежде. Заметим, что вторая и третья компоненты состояния играют роль буфера, через который проходят задерживаемые символы. Поскольку буфер рассчитан на два элемента, он обеспечивает задержку каждого символа на то время, пока машина проходит две позиции на ленте.

4. $\delta([q_1, A_1, A_2], B) = ([q_1, A_2, B], A_1, R)$ для $A_1, A_2, A_3 \in \Gamma \setminus \{B, X\}$.

Когда на ленте виден пробел, запомненный во второй компоненте символ помещается на ленту.

5. $\delta([q_1, A_1, B], B) = ([q_2, B, B], A_1, L)$ для $A_1 \in \Gamma \setminus \{B, X\}$.

Последний запомненный символ из второй компоненты состояния помещается на ленту, после чего начинается движение влево с целью выйти на крайнюю правую свободную ячейку. Она помечена, как и все другие освобожденные ячейки, символом X.

6. $\delta([q_2, B, B], A) = ([q_2, B, B], A, L)$ для $A \in \Gamma \setminus \{B, X\}$.

Машина двигается влево, пока не выходит на символ X. После этого машина перейдет в состояние, которое, как мы предполагаем, существует в множестве состояний Q, и возобновит другие свои действия.

6.2.5. Моделирование. Пусть B — автомат, который с входной цепочкой w последовательно проходит конфигурации $C_1, C_2, \ldots, C_n$. Неформально, мы говорим, что автомат A *моделирует* автомат B, если автомат A с входной цепочкой w проходит последовательно конфигурации, представляющие $C_1, C_2, \ldots, C_n$.

Возможно, что автомат A будет входить в другие конфигурации в промежутках между конфигурациями, представляющими конфигурации автомата B.

Понятие моделирования полезно при доказательстве, что автомат одного типа может распознавать язык, принимаемый автоматом некоторого другого типа. Чтобы автомат A моделировал автомат B, он должен быть способен, во-первых, по представлению конфигурации C_i вычислять представление C_{i+1}; во-вторых, автомат A должен определять, является ли конфигурация C_i автомата B принимающей. В этом случае A тоже должен принимать его собственную входную цепочку.

Попутно заметим, что для целей моделирования часто бывает удобно представлять конфигурацию машины Тьюринга в виде $\alpha qX\beta$, где α и β являются ленточными цепочками, X — символ ленты, q — состояние. В конфигурации $\alpha qX\beta$ состояние есть q, $\alpha X\beta$ — непустая часть ленты, а X — символ, сканируемый головкой ленты. Пример моделирования недетерминированной машины Тьюринга с помощью детерминированной приводится в § 6.4.

6.2.6. Диагонализация. Другое полезное понятие — *диагонализация*. Оно может быть использовано для того, чтобы показать, что существует язык, при-

нимаемый автоматом типа 2, который не принимается никаким автоматом типа 1. Диагонализация характеризуется несколькими отличительными чертами.

а) Должно существовать кодирование всех автоматов типа 1, входные символы которых выбираются из одного и того же алфавита Σ. Пример того, как это кодирование может быть сделано, приводится в следующей главе.

б) Должен строиться автомат A типа 2 с входными цепочками из Σ^*. Входная цепочка автомата A трактуется как кодирование некоторого автомата B типа 1 с его собственной входной цепочкой.

в) Автомат A должен моделировать автомат B и определять, принимает B свою собственную входную цепочку или нет. Если автомат B принимает входную цепочку, то автомат A не принимает, и наоборот.

г) Всегда истинно, что язык, принимаемый автоматом A типа 2, не принимается никаким автоматом типа 1. Действительно, предположим, что B является таким автоматом типа 1, который принимает язык, принимаемый автоматом A. Автомат B имеет кодирование $w \in \Sigma^*$. Предположим, что автомат B принимает цепочку w. Тогда автомат A не принимает цепочку w, и наоборот. В любом случае, автоматы A и B не могут принимать один и тот же язык.

Было бы уместно дать следующее разумное объяснение термина “диагонализация”. Можно перенумеровать все слова в множестве Σ^* следующим образом. Взять сначала самые короткие, и среди слов равной длины использовать некоторый лексикографический порядок. Затем мы можем занумеровать автоматы типа 1 согласно номерам, приписанным их кодам. Автомат A принимает i-е слово тогда и только тогда, когда i-е слово не принимается i-м автоматом. Вообразим бесконечную матрицу, элемент которой (i, j) есть 1, если i-й автомат принимает j-е слово, и 0 — в противном случае. Автомат A принимает i-е слово, когда элемент (i, i) есть 0. Отсюда и термин “диагонализация”.

6.2.7. Подпрограммы. Одна машина Тьюринга может быть “подпрограммой” другой машины Тьюринга при весьма общих условиях. Если T_1 должна быть подпрограммой для T_2, мы требуем, чтобы их состояния не пересекались. Но с состояниями других подпрограмм состояния T_1 могут пересекаться.

Чтобы “вызвать” T_1, машина T_2 входит в начальное состояние машины T_1. Правила машины T_1 являются частью правил машины T_2. Кроме того, из состояния остановки машины T_1, машина T_2 входит в свое собственное состояние и продолжает работу.

Пример 6.6. Опишем неформально машину Тьюринга T_3, которая вычисляет $n!$ Именно: машина, запущенная с входной цепочкой вида 01^n0, должна заканчивать работу с кодом вида $0^{n+2}1^{n!}$ на непустой части ее ленты.

Машина T_3 использует машину T_2 в качестве подпрограммы, которая выполняет умножение. Именно: машина T_2, запущенная с входной цепочкой вида

$01^i0^j1^k$ на ее ленте и ленточной головкой на крайнем левом нуле блока из j нулей, останавливается с результатом вида $01^i0^j1^{ik}$.

Из своей начальной конфигурации машина T_3, двигаясь вправо, проходит блок единиц и последний нуль, печатает 1 и движется влево. В этот момент на ленте машины T_3 находится цепочка 01^n01. Далее машина T_3 вызывает подпрограмму T_2. Когда управление вернется к машине T_3, ее лента будет содержать цепочку 01^n01^n0. Затем из своего текущего состояния машина T_3 проверяет, остается ли в первом блоке единиц, по крайней мере, три единицы. Если это так, она изменяет крайнюю правую единицу в этом блоке на нуль и возвращается в состояние, из которого она вызывает подпрограмму T_2.

После второго вызова подпрограммы T_2 на ленте машины T_3 появится цепочка $01^{n-1}001^{n(n-1)}0$. После третьего вызова на ленте будет последовательность $01^{n-2}0001^{n(n-1)(n-2)}0$, и вызовы T_2 будут повторяться до тех пор, пока после $(n-1)$-го вызова на ленте не появится цепочка $0110^{n-1}0001^{n(n-1)(n-2)\ldots 2}0$. В это время первый блок единиц имеет меньше трех 1, так что машина T_3 изменяет их на нули и останавливается при состоянии ленты $0^{n+2}1^{n(n-1)(n-2)\ldots 2}0 = 0^{n+2}1^{n!}0$.

В свою очередь, машина T_2 сама использует машину T_1 в качестве своей подпрограммы, которая добавляет первый блок единиц, не изменяя его, ко второму. При построении машины T_1 пригодится метод отметки посредством символов "отключения проверки". Машина T_2 вызывает подпрограмму T_1 один раз для каждой единицы в первом блоке, отключая ее проверку. Таким образом осуществляется умножение.

§ 6.3. Машина Тьюринга как процедура

До сих пор мы определяли машину Тьюринга как распознающее устройство. Но можно рассматривать машину Тьюринга и как процедуру. Например, если мы желаем описать процедуру для определения того, является ли число простым, мы могли бы построить машину Тьюринга, которая принимает множество всех простых чисел. Рассматриваем мы эту машину в данном случае как распознаватель или как процедуру — дело вкуса.

Если желательно рассмотреть процедуру манипуляции над строками символов, то можно свести ее к проблеме распознавания при помощи построения другой машины Тьюринга, которая принимает пары строк, разделенных специальным символом. Эта машина принимает данную пару точно тогда, когда процедура превратила бы первую строку в паре во вторую и остановилась. Мы оставим доказательство того факта, что по данной процедуре можно найти соответствующий распознаватель и наоборот, в качестве упражнения для читателя. Большинство необходимых для этого идей содержит §1.2.

Машина Тьюринга в примере 6.1 используется как распознаватель. Заметим, что независимо от того, какова входная цепочка, эта машина со временем дос-

тигнет условия, при котором для состояния конечного управления и сканируемого символа функция δ не определена. В таком случае машина Тьюринга останавливается и никакие дальнейшие ее движения невозможны. Если язык принимается машиной Тьюринга, которая останавливается на всех входных цепочках, то говорят, что язык *рекурсивен*.

Следует подчеркнуть, что существуют языки, которые принимаются машинами Тьюринга, не останавливающимися для некоторых цепочек, не содержащихся в языке, но которые не принимаются никакими машинами Тьюринга, останавливающимися на всех входных цепочках. Язык, который может быть распознан некоторой машиной Тьюринга, называется *рекурсивно перечислимым множеством* (recursively enumerable set — res). В следующей главе будет показано, что рекурсивно перечислимые множества являются в точности языками, порождаемыми грамматиками типа 0.

Когда машина Тьюринга рассматривается как процедура и оказывается, что она останавливается для всех входных цепочек, то говорят, что такая процедура есть *алгоритм*.

Есть процедуры, для которых не существует никакого соответствующего алгоритма. Примером их является процедура для определения, порождает ли контекстно-зависимая грамматика (csg), по крайней мере, одну терминальную цепочку. Можно построить машину Тьюринга, которая по заданной csg будет порождать все возможные терминальные цепочки в некотором лексикографическом порядке. К каждой цепочке эта машина Тьюринга применяет алгоритм, данный в гл. 2, чтобы увидеть, порождается ли данная цепочка грамматикой. Если эта грамматика порождает хотя бы одно слово, машина найдет его и остановится в конечном состоянии. Но если язык, порождаемый этой грамматикой, пуст, то машина будет продолжать порождать слова и проверять их вечно. Имеет место факт, что не существует машины Тьюринга, которая останавливается на каждой входной цепочке и определяет для каждой csg, является ли язык, порождаемый этой грамматикой, пустым. Другими словами, проблема распознавания непустоты контекстно-зависимого языка алгоритмически неразрешима.

В дополнение к рассмотрению машины Тьюринга как распознающего устройства или процедуры мы можем рассматривать ее как *средство определения функций*. Пусть $f(n)$ — функция, отображающая положительные целые в положительные целые и пусть $T = (Q, \Sigma, \Gamma, \delta, q_0, F)$ — машина Тьюринга. Если для каждого целого n имеет место $(q_0, 1^n, 1) \overset{*}{\underset{T}{\vdash}} (p, 1^{f(n)}, 1)$ для $p \in F$, то говорят, что машина вычисляет функцию $f(n)$.

Если некоторая машина Тьюринга T вычисляет функцию $f(n)$ для каждого n, то говорят, что функция $f(n)$ *рекурсивна*.

Если $f(n)$ определена для не всех n, то говорят, что $f(n)$ — *частичная функция.*

Если некоторая машина Тьюринга T вычисляет функцию $f(n)$ всякий раз, как $f(n)$ определена, но может не остановиться для тех n, для которых $f(n)$ не определена, то $f(n)$ — *частично рекурсивная функция*.

§ 6.4. Модификации машин Тьюринга

Одна из причин, по которой машины Тьюринга принимаются в качестве общей модели вычисления, состоит в том, что модель, с которой мы имеем дело, инвариантна по отношению ко многим модификациям, которые, казалось бы, увеличивают вычислительную мощность устройства. В этом параграфе даются доказательства некоторых из этих теорем об эквивалентности.

6.4.1. Машина Тьюринга с бесконечной лентой в обе стороны обозначается как и первоначальная модель: $T = (Q, \Sigma, \Gamma, \delta, q_0, F)$. Как подразумевает ее название — лента бесконечна не только вправо, но и влево. Конфигурация такого устройства, как и прежде, есть (q, α, i), где $q \in Q$ — состояние, $\alpha \in \Gamma^+$ — непустая строка символов, но i (и это не обычно) — *положение головки ленты относительно левого конца строки* α. Другими словами, $i = 1$, если машина сканирует самый левый символ цепочки α; $i = 2$, если она сканирует второй символ, и т.д. Предполагается, что имеется бесконечно много пустых ячеек не только справа, но и слева от цепочки α. Таким образом, возможно, что $i = 0$, и в этом случае машина сканирует пробел непосредственно слева от цепочки α.

Отношение $\vdash_T$, связывающее две конфигурации, из которых вторая получается из первой при помощи единственного движения, определяется как для первоначальной модели со следующими исключениями для $i \le 1$:

a) если $\delta(q, X) = (p, Y, L)$, то $(q, X\alpha, 1) \vdash_T (p, Y\alpha, 0)$;

b) если $\delta(q, B) = (p, Y, R)$, то $(q, \alpha, 0) \vdash_T (p, Y\alpha, 2)$;

c) если $\delta(q, B) = (p, Y, L)$, то $(q, \alpha, 0) \vdash_T (p, Y\alpha, 0)$.

Здесь B — пробел, а $Y \ne B$. Начальной конфигурацией является $(q_0, w, 1)$. В отличие от первоначальной модели нет никакого левого конца ленты, который "выпадает", так что она может двигаться влево сколь угодно далеко.

Степень, транзитивное замыкание и рефлексивно-транзитивное замыкание отношения $\vdash_T$ определяются аналогично.

Теорема 6.1. *Если язык L распознается машиной Тьюринга* (*Tm*) *с бесконечной в обе стороны лентой, то он распознается Tm с полубесконечной лентой.*

Доказательство. Пусть $T_2 = (Q_2, \Sigma_2, \Gamma_2, \delta_2, q_2, F_2)$ — Tm с лентой, бесконечной в обе стороны. Мы построим T_1 — машину Тьюринга, моделирующую машину T_2 и имеющую ленту, которая бесконечна только вправо. Эта лента имеет две дорожки. На одной представляются ячейки ленты машины T_2, расположенные вправо от начального положения головки, включая и саму ячейку, сканируемую вначале. На другой — ячейки, расположенные слева от начальной в порядке их удаления от начальной позиции. Если мы припишем начальной ячейке машины T_2 номер 0, ячейкам, которые справа, — номера 1, 2,..., а тем ячейкам, которые слева, припишем номера –1, –2, ... , то связь между лентами машин T_2

(рис. 6.3, *а*) и T_1 (рис. 6.3, *б*) может быть представлена так, как показано на этом рисунке.

а

A_0	A_1	A_2	A_3	A_4	…

б

A_0	A_1	A_2	A_3	A_4	…
¢	A_{-1}	A_{-2}	A_{-3}	A_{-4}	…

Рис. 6.3.

Первая ячейка на ленте машины T_1 будет содержать на нижней дорожке символ ¢, указывающий, что это — самая левая ячейка. Конечное управление машины T_1 будет содержать информацию относительно того, сканирует ли машина T_2 символ, находящийся на верхней или на нижней дорожке.

Очевидно, что машина T_1, моделирующая машину T_2, может быть построена. Когда машина T_2 находится справа от начальной позиции, машина T_1 работает на верхней дорожке. Когда машина T_2 находится слева от начальной позиции, машина T_1 работает на нижней дорожке, двигаясь в направлении, противоположном направлению движения машины T_2. Входные символы машины T_1 — символы с пробелом на нижней дорожке и входным символом машины T_2 на верхней дорожке. Такой символ идентифицируется с соответствующим входным символом машины T_2.

Теперь выполним формальное построение $T_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_1, F_1)$.

Положим $Q_1 = \{q_1\} \cup \{[q, D] \mid q \in Q_2, D \in \{U, L\}\}$. Здесь вторая компонента состояния указывает, работает ли машина T_1 на верхней или нижней дорожке[9].

Ленточные символы $\Gamma_1 = \{[X, Y] \mid X \in \Gamma_2, Y \in \Gamma_2 \cup \{¢\}, ¢ \notin \Gamma_2\}$. Если B — пробел на ленте машины T_2, то $[B, B]$ — пробел на ленте машины T_1.

Входные символы $\Sigma_1 = \{[a, B] \mid a \in \Sigma_2\}$. Мы идентифицируем a с $[a, B]$.

Конечные состояния $F_1 = \{[q, D] \mid q \in F_2, D \in \{U, L\}\}$.

Функцию δ_1 определим следующим образом:

1. Для любого $a \in \Sigma_2$ полагаем

$\delta_1(q_1, [a, B]) = ([q, U], [X, ¢], R)$, если $\delta_2(q_2, a) = (q, X, R)$.

Если первое движение машины T_2 происходит вправо, то машина T_1 печатает ¢ на нижней дорожке, чтобы отметить левый край ленты, устанавливает вторую компоненту состояния на U и двигается вправо. Первая компонента состояния машины T_1 фиксирует состояние машины T_2. При этом машина T_1 печатает символ X на верхней дорожке, если машина T_2 печатает его на своей ленте.

2. Для любого $a \in \Sigma_2$ полагаем

$\delta_1(q_1, [a, B]) = ([q, L], [X, ¢], R)$, если $\delta_2(q_2, a) = (q, X, L)$.

[9] Здесь символ L обозначает lower (нижняя) и его не следует путать с L в составе значений δ, где L обозначает left (влево).

Если первое движение машины T_2 происходит влево, то машина T_1 печатает $\not\subset$ на нижней дорожке, чтобы отметить левый край ленты, устанавливает вторую компоненту состояния на L и двигается вправо. Первая компонента состояния T_1 фиксирует состояние T_2. При этом машина T_1 печатает символ X на верхней дорожке, если машина T_2 печатает его на своей ленте.

3. Для любого $[X, Y] \in \Gamma_1$ с $Y \neq \not\subset$ и $D \in \{L, R\}$ полагаем

$$\delta_1([q, U], [X, Y]) = ([p, U], [Z, Y], D), \text{ если } \delta_2(q, X) = (p, Z, D).$$

4. Для любого $[X, Y] \in \Gamma_1$ с $Y \neq \not\subset$ и $D \in \{L, R\}$ полагаем

$$\delta_1([q, L], [X, Y]) = ([p, L], [X, Z], D), \text{ если } \delta_2(q, Y) = (p, Z, \overline{D}).$$

Если $D = L$, то $\overline{D} = R$. Если $D = R$, то $\overline{D} = L$.

Машина T_1 моделирует машину T_2 на своей нижней дорожке. Направление движения головки машины T_1 противоположно направлению движения машины T_2.

5. Полагаем, что

$$\delta_1([q, U], [X, \not\subset]) = \delta_1([q, L], [X, \not\subset]) = ([p, E], [Y, \not\subset], R), \text{ если } \delta_2(q, X) = (p, Y, D);$$

$E = U$, если $D = R$; $E = L$, если $D = L$.

Машина T_1 моделирует движение машины T_2 на ячейке, первоначально сканированной машиной T_2. Затем машина T_1 работает на верхней или нижней дорожке в зависимости от направления, в котором движется машина T_2. В этой позиции машина T_1 будет всегда двигаться вправо.

Доказательство того, что обе машины принимают один и тот же язык, предоставляется читателю в качестве упражнения.

6.4.2. Многоленточная машина Тьюринга состоит из конечного управления с k ленточными головками, по одной на каждой ленте (рис. 6.4).

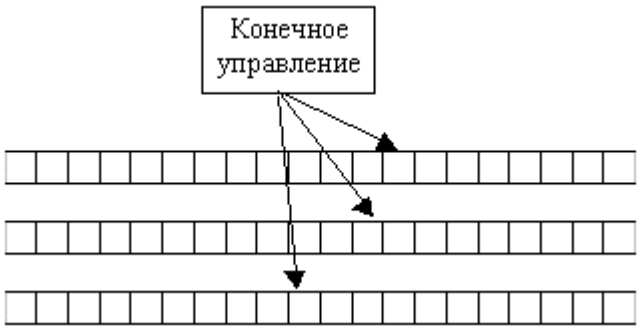

Рис. 6.4.

Каждая лента бесконечна в обоих направлениях. При одном движении, зависящем от состояния конечного управления и сканируемого символа каждой из ленточных головок, машина может:

1) изменить состояние;

2) напечатать новый символ на каждой из сканируемых ячеек;

3) передвинуть каждую из ее ленточных головок независимо друг от друга на одну ячейку влево, вправо или оставить ее на том же месте.

Сначала входная цепочка имеется только на первой ленте, а все другие ленты пусты. Мы не будем определять это устройство более формально, предоставляя это читателю.

Теорема 6.2. *Если язык L принимается многоленточной машиной Тьюринга, то он принимается одноленточной машиной Тьюринга.*

Доказательство. Пусть язык L принимается машиной Тьюринга T_1 с k лентами. Построим одноленточную машину Тьюринга T_2 с $2k$ дорожками, по две для каждой из лент машины T_1. На одной дорожке записывается содержимое соответствующей ленты машины T_1, а другая — пустая, за исключением маркера в ячейке, содержащей символ и сканируемой соответствующей головкой машины T_1. Такое устройство для моделирования трех лент посредством одной иллюстрируется рис. 6.5.

Конечное управление машины T_2 запоминает, какие маркеры головок машины T_1 находятся слева, а какие — справа от головки T_2. Состояния машины T_1 тоже запоминаются в конечном управлении машины T_2.

Головка 1		×					
Лента 1	A_1	A_2	…		…	A_m	
Головка 2				×			
Лента 2	B_1	B_2	…		…	B_m	
Головка 3	×						
Лента 3	C_1	C_2	…		…	C_m	

Рис. 6.5.

Чтобы моделировать движение машины T_1, машина T_2 должна посетить каждую ячейку с маркером головки, регистрируя по очереди символ, сканируемый соответствующей головкой T_1. Когда машина T_2 проходит через маркер головки, она должна уточнять направление, в котором следует искать этот маркер. После сбора всей необходимой информации машина T_2 определяет движение машины T_1. Затем машина T_2 посещает по очереди каждый из маркеров головок снова, изменяя маркированные ячейки и сдвигая маркеры на одну ячейку, если необходимо. Конечно, если новое состояние является принимающим, то машина T_2 принимает входную цепочку.

Пример 6.7. Посмотрим, насколько легче многоленточной машине Тьюринга распознать язык $L = \{ww^R \mid w \in \{0,1\}^*\}$, чем одноленточной.

Чтобы распознать язык L на одноленточной машине Тьюринга, ее головка ленты должна двигаться вперед и назад по ленте, отмечая и сравнивая символы с обоих концов. Это подобно процессу из примера 6.4.

Чтобы распознать язык L с помощью двухленточной машины Тьюринга, входная цепочка копируется на вторую ленту. Затем цепочка на одной ленте

сравнивается с ее копией на другой ленте в обратном порядке, и ее длина проверяется на четность.

Заметим, что для распознавания языка L однолентоточная машина делает $O(n^2)$ движений, где n — длина входной цепочки, тогда как двухленточной машине достаточно совершить только $O(n)$ движений.

6.4.3. Недетерминированная машина Тьюринга есть устройство с конечным управлением и одной бесконечной в обе стороны лентой. Для данного состояния и ленточного символа, сканируемого головкой ленты, машина имеет несколько вариантов для следующего движения. Каждый вариант состоит из нового состояния, ленточного символа, который печатается, и направления движения головки. Недетерминированная машина Тьюринга принимает входную цепочку, если какая-нибудь последовательность вариантов движений приводит к принимающему состоянию.

Теорема 6.3. *Если язык L принимается недетерминированной машиной Тьюринга T_1, то он принимается некоторой детерминированной машиной Тьюринга T_2.*

Доказательство. Для любого состояния и ленточного символа машины T_1 имеется конечное число вариантов для выбора следующего движения. Варианты могут быть занумерованы числами 1, 2, ... Пусть r — максимальное число вариантов для любой пары состояние — ленточный символ. Тогда любая последовательность вариантов движений конечной длины может быть представлена последовательностью цифр от 1 до r. Не все такие последовательности могут представлять варианты движений, поскольку в некоторых конфигурациях вариантов может быть меньше, чем r.

Можно построить детерминированную машину Тьюринга T_2, моделирующую машину T_1. Снабдим ее тремя лентами. Первая будет содержать входную цепочку. На второй машина T_2 будет систематически генерировать последовательность цифр от 1 до r. Конкретно: последовательности будут генерироваться, начиная с самой короткой. Среди последовательностей одинаковой длины они генерируются в числовом порядке.

Для каждой последовательности, сгенерированной на ленте 2, машина T_2 копирует вход на ленту 3 и затем моделирует машину T_1 на ленте 3, используя последовательность ленты 2 для того, чтобы диктовать движения машине T_1.

Если машина T_1 входит в принимающее состояние, то машина T_2 также принимает. Если имеется последовательность вариантов, ведущая к приему, то она в конце концов будет сгенерирована на ленте 2. Будучи смоделирована, машина T_2 будет принимать входную цепочку. Но если никакая последовательность вариантов движений машины T_1 не ведет к приему входной цепочки, то машина T_2 не примет ее.

Заметим, что это доказательство можно обобщить, чтобы показать, как моделировать недетерминированную многоленточную машину Тьюринга обычной моделью машины Тьюринга.

6.4.4. Двумерная машина Тьюринга является еще одной модификацией машины Тьюринга, которая не увеличивает ее мощности. Это устройство состоит из обычного конечного управления, но лента разбита на бесконечное число ячеек, расположенных в двух измерениях. В зависимости от состояния и сканируемого символа устройство изменяет состояние, печатает новый символ и передвигает ленточную головку в одном из четырех направлений. Первоначально входная цепочка находится на одной строке, а головка находится на левом конце вводной цепочки.

В любое время только конечное число строк имеет какие-нибудь непустые символы на них, и каждая из этих строк имеет только конечное число непустых символов.

Рассмотрим например, конфигурацию ленты, показанную на рис. 6.6, *а*. Мы можем очертить прямоугольник вокруг непустых символов, как показано на этом рисунке, и этот прямоугольник можно записать строка за строкой на одномерной ленте, как показано на рис. 6.6, *б*.

а

B	B	B	a_1	B	B	B
B	B	a_2	a_3	a_4	a_5	B
a_6	a_7	a_8	a_9	B	a_{10}	B
B	a_{11}	a_{12}	a_{13}	B	a_{14}	a_{15}
B	B	a_{16}	a_{17}	B	B	B

б

$*BBBa_1BBB*BBa_2a_3a_4a_5B*a_6a_7a_8a_9Ba_{10}B*Ba_{11}a_{12}a_{13}Ba_{14}a_{15}*BBa_{16}a_{17}BBB*$

Рис. 6.6.

Символы * разделяют строки. Один из символов помечается как сканируемый головкой. Для пометки можно использовать дополнительную дорожку ленты. Если при данном движении головка остается в пределах представленного прямоугольника, то подогнать положение головки нетрудно. Если головка выходит за границу прямоугольника при движении в вертикальном направлении, добавляют еще одну строку пробелов к левому или правому концу линейного представления. Если головка покидает прямоугольник через правую или левую границу, длина каждой представленной строки должна быть увеличена на единицу. При этом может пригодиться метод “сдвига”.

Описанный подход легко обобщить на n-мерные ленты.

6.4.5. Машина Тьюринга с входной лентой только для чтения и одной или несколькими лентами памяти для записи/чтения также заслуживает упоминания. Ее движение зависит от сканируемого входного символа, но она не может печатать на входной ленте. Обычно считается, что входная лента имеет концевые маркеры, так что головка входной ленты может всегда оставаться на входной цепочке, границы которой она не может сама отмечать.

Если входная головка может двигаться в двух направлениях, то устройство называется *машиной Тьюринга типа off-line*. Если входная головка никогда не движется влево, то это *машина Тьюринга типа on-line*. Ясно, что машины этих двух типов являются вариантами многоленточных машин Тьюринга. Они могут моделировать любую многоленточную машину Тьюринга.

§ 6.5. Ограниченные машины Тьюринга, эквивалентные основной модели

До сих пор рассматривались обобщения основной модели машины Тьюринга. Как мы видели, эти обобщения не увеличивают вычислительную мощность этой модели.

Данную главу заключим обсуждением некоторых моделей, которые с первого взгляда могут показаться менее мощными, чем обычные машины Тьюринга, но на самом деле имеют точно такую же мощность. По большей части эти модели будут вариациями базисного магазинного автомата, рассмотренного в гл. 5. Попутно отметим, что магазинный автомат можно рассматривать как недетерминированную машину Тьюринга с двумя лентами: одной только для чтения, причем ее входная головка не может двигаться влево, и другой — лентой памяти с весьма специфическим ограничением для ее ленточной головки. Всякий раз, как головка ленты памяти движется влево, она должна печатать пробел. Таким образом, лента памяти справа от головки — всегда пустая. Строго говоря, машине Тьюринга запрещено печатать настоящий пробел. Взамен можно было бы ввести другой специальный символ с теми же самыми правилами, какие имеются для пробела. Предоставим читателю убедиться в том, что такая модель эквивалентна магазинному автомату, введенному в гл. 5.

6.5.1. Детерминированная машина с двумя магазинными лентами — это детерминированная машина Тьюринга с входной цепочкой только для чтения и двумя магазинными лентами памяти. Если головка магазинной ленты движется влево, то она печатает “пробел”.

Лемма 6.1. *Произвольная одноленточная машина Тьюринга может быть смоделирована детерминированной машиной с двумя магазинными лентами*.

Доказательство. Достаточно убедиться в том, что символы слева от головки машины Тьюринга, которая моделируется, могут запоминаться на одной магазинной ленте, а те, что справа, — на другой.

6.5.2. Машина со счетчиками есть машина Тьюринга, алфавиты лент памяти которой содержат только два символа, например Z и B. Кроме того, символ Z появляется в ячейке, первоначально сканируемой ленточной головкой, и никогда не может появиться ни в какой другой. Число i можно запомнить путем передвижения головки ленты на i ячеек вправо от Z. Предполагается, что машины этого типа могут печатать пробел. Запомненное число может быть увеличено или уменьшено передвижением головки ленты вправо или влево.

Пример машины со счетчиком показан на рис. 6.7. Символы ¢ и $ обычно используются в качестве концевых маркеров на входной ленте. Здесь Z — непустой символ.

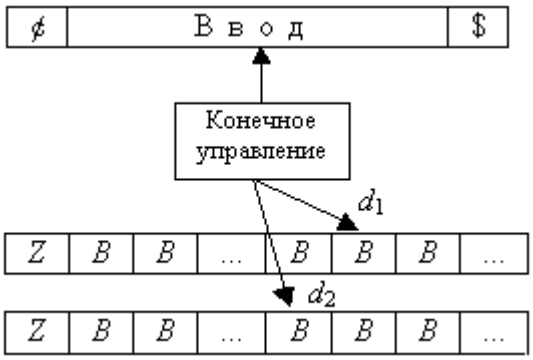

Рис. 6.7.

Конфигурация машины со счетчиками может быть описана при помощи состояния, положения входной головки и расстояний головок лент памяти от символа Z (на рис. 6.7 они обозначены символами d_1 и d_2). Назовем эти расстояния *отсчетами* на лентах. Машина со счетчиками может фактически только запоминать отсчет на каждой ленте и указывать, является ли этот отсчет нулевым.

Лемма 6.2. *Машина с четырьмя счетчиками может моделировать произвольную машину Тьюринга*.

Доказательство. Согласно лемме 6.1 достаточно показать, что две ленты со счетчиками могут моделировать одну магазинную ленту.

Пусть магазинная лента имеет $k-1$ непустых ленточных символов Z_1, Z_2, ..., Z_{k-1}. Тогда мы можем однозначно представить магазинную ленту $Z_{i_1}Z_{i_2}...Z_{i_m}$ при помощи "счетчика" $j = i_m + k\, i_{m-1} + k^2\, i_{m-2} + ... + k^{m-1} i_1$. Предположим, что j запоминается на одном счетчике, т.е. головка ленты находится на j ячеек правее непустого символа. Пусть головка второго счетчика также находится на непустом символе.

Предположим, что символ Z_r печатается на вершине (правом конце) магазинной ленты $Z_{i_1}Z_{i_2}...Z_{i_m}$. Счетчик, связанный с $Z_{i_1}Z_{i_2}...Z_{i_m}Z_r$, равен $kj + r$. Чтобы установить этот новый счетчик, машина со счетчиком повторно сдвигает головку первого счетчика на одну ячейку влево, а головку второго — на k ячеек вправо. Когда головка первого счетчика достигнет непустого символа, второй счетчик будет представлять kj. Добавить r к этому счетчику просто. Наоборот, чтобы удалить верхний символ Z_{i_m} магазина, необходимо заменить значение счетчика j на целую часть j / k. Для этого надо повторно продвигать влево головку первого счетчика на k ячеек, и после каждого такого продвижения головки второго счетчика продвигать влево только на одну ячейку. Если в очередной раз окажется невозможно продвинуть головку первого счетчика ровно на k ячеек влево из-за того, что непустая ячейка будет достигнута раньше, чем закончится продвижение на k ячеек, то в этот момент на втором счетчике образуется требуемое значение.

Чтобы закончить описание моделирования, остается рассказать, как узнать, какой символ находится на вершине магазинной ленты, моделируемой двумя счетчиками. Если установлено значение j на одном счетчике, то машина может скопировать j в другой счетчик, вычисляя j по модулю k в своем конечном управлении. Заметим, что j по модулю k это и есть i_m.

Теорема 6.4. *Машина с двумя счетчиками может моделировать произвольную машину Тьюринга*.

Доказательство. Согласно лемме 6.2 достаточно показать, как моделировать четыре счетчика двумя. Пусть четыре счетчика имеют отсчеты i, j, k и m. Все четыре значения можно установить на одном счетчике в виде числа $n = 2^i3^j5^k7^m$. Поскольку 2, 3, 5 и 7 — простые числа, то значения i, j, k и m могут быть однозначно восстановлены из n.

Чтобы увеличить i, j, k или m на единицу, достаточно умножить n на 2, 3, 5 или 7 соответственно.

Если мы имеем второй счетчик, установленный на нуль, то можем сдвигать головку этого счетчика на 2, 3, 5 или 7 ячеек вправо каждый раз, как головка первого счетчика продвигается на одну ячейку влево. Когда первый счетчик достигнет нулевого отсчета, второй будет содержать отсчеты $2n$, $3n$, $5n$ или $7n$ соответственно.

Чтобы уменьшить i, j, k или m на единицу, достаточно при помощи подобного процесса разделить n на 2, 3, 5 или 7 соответственно.

Мы должны также показать, как машина с двумя счетчиками может определить следующее движение машины с четырьмя счетчиками. Входная головка машины с двумя счетчиками будет всегда в той же самой точке на ее входной ленте, в какой была бы входная головка машины с четырьмя счетчиками. Состояние машины с четырьмя счетчиками может быть запомнено в конечном управлении двухсчетчиковой машины. Следовательно, чтобы определить движение четырехсчетчиковой машины, двухсчетчиковая должна только определить, какие отсчеты из i, j, k, m равны нулю. Передавая n из одного счетчика в другой, конечное управление двухсчетчиковой машины может определить, делится ли n на 2, 3, 5, 7 или какое-нибудь их произведение.

Теорема 6.5. *Каждая машина Тьюринга может быть смоделирована машиной Тьюринга с входной лентой только для чтения и лентой памяти с двумя символами* (*пробел и другой символ*) *при условии*, *что машина может печатать пробелы*.

Доказательство. Большая часть доказательства будет оставлена читателю. Прием состоит в том, чтобы закодировать каждый из k символов памяти при помощи r бинарных символов, где $2^r \geq k$. Головка ленты машины Тьюринга может посетить каждый из r бинарных символов, представляющих первоначальный символ, чтобы определить, каким он был.

Замечательно, что может быть доказана теорема более сильная, чем теорема 6.5.

Теорема 6.6. *Каждая машина Тьюринга может моделироваться машиной Тьюринга с входной лентой только для чтения и лентой памяти с двумя символами*: 0 (*пробел*) *и* 1. *Машина Тьюринга может печатать* 0 *или* 1, *где был найден 0, но не может печатать 0 там, где находилась* 1.

Доказательство мы оставляем читателю. Прием состоит в том, чтобы смоделировать последовательные конфигурации исходной машины Тьюринга на ленте новой машины. Символы ленты, конечно, кодируются в бинарных символах. Каждая конфигурация копируется и попутно производятся необходимые изменения, отражающие движение исходной машины.

Помимо бинарного кодирования первоначального символа машина Тьюринга, моделирующая исходную, нуждается в ячейках, чтобы указывать (а) позицию головки в конфигурации, которая копируется, и (б) что бинарное представление символа уже скопировано.

Глава 7

МАШИНЫ ТЬЮРИНГА: ПРОБЛЕМА ОСТАНОВКИ, ЯЗЫКИ ТИПА 0

§ 7.1. Универсальная машина Тьюринга

В этой главе мы покажем, что существует машина Тьюринга U, которая по заданному коду произвольной машины Тьюринга T и кодированию входной цепочки x будет моделировать поведение машины T с входной цепочкой x. Такая машина U называется *универсальной машиной Тьюринга*. Ее можно рассматривать как вычислительную машину общего назначения, которая достаточно мощна для того, чтобы моделировать любую вычислительную машину, включая саму себя.

Мы покажем также, что не существует алгоритма (т.е. машины Тьюринга, которая останавливается на всех входных цепочках), который мог бы определить для произвольной машины Тьюринга T и произвольной ее входной цепочки x, остановится ли когда-нибудь машина T с входной цепочкой x. Этот отрицательный результат интенсивно используется как аргумент для того, чтобы показать, что многие проблемы, относящиеся к различным классам языков, являются рекурсивно (т.е. алгоритмически) неразрешимыми.

Будет также показано, что имеются рекурсивно перечислимые множества, которые не являются рекурсивными. Другими словами, есть множества, которые распознаются машинами Тьюринга, но не такими, которые останавливаются для всех входных цепочек.

Наконец, будет доказана основная теорема об эквивалентности языков типа 0 и множеств, распознаваемых машинами Тьюринга.

Покажем, что универсальная машина Тьюринга существует путем действительного ее построения.

Прежде всего мы должны условиться относительно кодирования машин Тьюринга и кодирования их входных цепочек. Поскольку машина Тьюринга T_1 может иметь любое число допустимых символов ленты, мы предполагаем, что все они будут кодироваться при помощи символов 0 и 1. Очевидно, что для каждой Tm T_1 существует Tm T_2 с ленточными символами 0 и 1 и одним дополнительным символом ленты B (пробел), которая принимает точно те строки из множества $\{0, 1\}^*$, которые являются кодами слов, принимаемых машиной T_1. Принимая это во внимание, достаточно спроектировать универсальную машину Тьюринга для машин Тьюринга с одинаковыми ленточными алфавитами $\{0, 1, B\}$.

Машина Тьюринга с тремя допустимыми ленточными символами может быть полностью определена при помощи лл. 7.1.

Табл. 7.1

Состояние	Входной символ		
	B	0	1
1	—	—	2, 0, *R*
2	3, 1, *L*	3, 1, *L*	2, 1, *R*
3	4, 0, *R*	4, 0, *R*	3, 1, *L*
4	—	—	—

Поскольку машина Тьюринга может иметь произвольно большое число состояний и поскольку мы имеем только фиксированное число допустимых символов ленты, то кодируем состояния в виде 1, 11, 111, и т.д.

Один из способов закодировать таблицу состояний состоит в том, чтобы разметить некоторое число блоков, равное числу состояний, а затем разбить каждый блок на три подблока. Состоянию *i* будет соответствовать *i*-й блок, а три подблока будут относиться к входным символам *B*, 0 и 1 соответственно. Блоки будут отделяться двумя символами *c*, а подблоки одним символом *c*. Начало и конец таблицы будет отмечаться тремя символами *c*.

Если в машине Тьюринга *T*, которая кодируется, $\delta(i, a) = (j, b, D)$, то подблок, соответствующий состоянию i и входному символу a, будет содержать j единиц, за которыми следует символ $D \in \{L, R\}$, а за ним $b \in \{0, 1\}$. Если $\delta(i, a)$ не определено, то соответствующий подблок будет содержать единственный нуль. Таким образом, кодировка табл. 7.1 оказалась бы такой, как приводимая ниже запись:

ccc 0c0c11R0cc
111L1c111L1c11Rcc
1111R0c1111R0c111L1cc
0c0c0ccc

Код 11*R*0 в подблоке, соответствующем состоянию 1 и входному символу 1, означает то, что машина *T*, будучи в состоянии 1 и сканируя символ 1, будет заменять 1 на 0, двигаться вправо и входить в состояние 2.

Любое состояние, в котором для всех трех допустимых символов ленты переходные состояния не определены, интерпретируется как принимающее состояние. Таково состояние 4 в табл. 7.1. Предполагается, что после приема входной цепочки машина не делает больше никаких движений.

В любом не принимающем состоянии, по крайней мере, для одного допустимого символа ленты следующее состояние должно быть определено.

В роли начального состояния всегда используется состояние 1.

Хотя мы использовали только пять символов, чтобы закодировать машину Тьюринга, показанную в табл. 7.1, наша универсальная машина Тьюринга будет использовать 12 символов ленты. Дополнительные символы появляются из того соображения, что она будет иметь двухдорожечную ленту. Нижняя дорожка будет использовать символы *c*, 0, 1, *L*, *R* и *B*, в то время как верхняя — символы *m* и *B*.

Табл.7.2

Сост.	*B*						*m*						Пояснения
	0	1	c	*L*	*R*	*B*	0	1	c	*L*	*R*	*B*	
A	Двигаться вправо					—	—	—	*B*, *R*	—	—	—	Найти маркер в области данных
B	Двигаться вправо					—	C_0, *L*	C_0, *L*	—	—	—	C_B, *L*	
C_B	Двигаться влево					—	—	—	$D_B, \binom{B}{c}, R$	—	—	—	Найти маркер в области таблицы состояний
C_0	Двигаться влево					—	—	—	$D_0, \binom{B}{c}, R$	—	—	—	
C_1	Двигаться влево					—	—	—	$D_1, \binom{B}{c}, R$	—	—	—	
D_B	*V*, *L*	$E, \binom{m}{1}, L$	—	—	—	—							Найти подблок, соотвествущий входному символу
D_0	*R*	*R*	D_B, *R*	*R*	*R*	—							
D_1	*R*	*R*	D_0, *R*	*R*	*R*	—							
E	*L*	*L*	*F*, *L*	*L*	*L*	—							Найти состояние 1 и отметить. В текстовом описании этот маркер — m_2
F	*E*, *L*	*E*, *L*	*G*, *L*	*E*, *L*	*E*, *L*	—							
G	*E*, *L*	*E*, *L*	*H*, *R*	*E*, *L*	*E*, *L*	—							
H	—	—	*I*, *R*	—	—	—							
I	—	—	$J, \binom{m}{c}, R$	—	—	—							
J	Двигаться вправо					—	—	$K_L, \binom{B}{1}, R$	—	—	—	—	Начало подпрограммы усновки следующего состояния

Сост.	*B*						*m*						Пояснения
	0	1	*c*	*L*	*R*	*B*	0	1	*c*	*L*	*R*	*B*	
K_L	—	$M_L, \binom{m}{1}, L$	—	T_L, R	T_R, R	—							
M_L	Двигаться влево					—	—	—	$N_L, \binom{B}{c}, R$	—	—	—	
N_L	R	R	M_L, R	R	R	—	—	N_R, R	—	—	—	—	m_2 *слева от* m_1
P_L	N_L, R	N_L, R	$S_L, \binom{m}{c}, R$	N_L, R	N_L, R	—	—	N_R, R	—	—	—	—	
S_L	Двигаться вправо					—	—	$K_L, \binom{B}{1}, R$	—	—	—	—	
K_R	—	$M_R, \binom{m}{1}, R$	—	T_L, R	T_R, R	—							
M_R	Двигаться вправо					—	—	—	$N_R, \binom{B}{c}, R$	—	—	—	
N_R	R	R	P_R, R	R	R	—							m_2 справа от m_1
P_R	N_R, R	N_R, R	$S_R, \binom{m}{c}, L$	N_R, R	N_R, R	—							
S_R	Двигаться влево					—	—	$K_N, \binom{B}{1}, R$	—	—	—	—	
T_L	T_{L-0}, R	T_{L-1}, R	—	—	—	—							Запоминание символа для печати
T_R	T_{L-0}, R	T_{L-1}, R	—	—	—	—							
T_{L-0}	Двигаться вправо					—	$U, \binom{B}{0}, L$	$U, \binom{B}{0}, L$	—	—	—	$U, \binom{B}{0}, L$	Найти маркер в области данных
T_{L-1}	Двигаться вправо					—	$U, \binom{B}{1}, L$	$U, \binom{B}{1}, L$	—	—	—	$U, \binom{B}{1}, L$	
T_{R-0}	Двигаться вправо					—	$U, \binom{B}{0}, R$	$U, \binom{B}{0}, R$	—	—	—	$U, \binom{B}{0}, R$	
T_{R-1}	Двигаться вправо					—	$U, \binom{B}{1}, R$	$U, \binom{B}{1}, R$	—	—	—	$U, \binom{B}{1}, R$	

Окончание табл. 7.2

Сост.	*B*						*m*						Пояснения
	0	1	*c*	*L*	*R*	*B*	0	1	*c*	*L*	*R*	*B*	
U	$C_0, \binom{m}{0}, L$	$C_1, \binom{m}{1}, L$	—	—	—	$C_B, \binom{m}{B}, L$							Установить маркер
V	*L*	*L*	*W*, *L*	*L*	*L*	—							Проверить, является ли состояние, в котором случилась остановка, принимающим
W	*V*, *L*	*V*, *L*	X_1, *R*	*V*, *L*	*V*, *L*	—							
X_1	—	—	X_2, *R*	—	—	—							
X_2	X_3, *R*	—	—	—	—	—							
X_3	—	—	X_4, *R*	—	—	—							
X_4	X_5, *R*	—	—	—	—	—							
X_5	—	—	X_6, *R*	—	—	—							
X_6	*Y*, *R*	—	—	—	—	—							
Y	—	—	—	—	—	—							Принять

Примечание. Элементы таблицы имеют следующие значения. Тройка из состояния, ленточного символа и *L* или *R* указывает следующее состояние, символ ленты, который печатать, и направление движения. Пара из состояния и *L* или *R* указывает следующее состояние и направление движения, при этом ленточный символ остается неизменным. Символы *L* или *R* и слова “Двигаться влево” или “Двигаться вправо” указывают, что машина Тьюринга движется влево или вправо, не изменяя состояния и символа ленты. Прочерк обозначает ситуацию, которая никогда не случается. Состояние *Y* является принимающим. Следовательно, никакое движение из состояния *Y* невозможно.

Теперь неформально опишем универсальную машину Тьюринга (U). Вход в нее, как сказано, устроен на двухдорожечной ленте. Нижняя дорожка будет содержать кодировку некоторой машины Тьюринга (T), за которой будет следовать цепочка из нулей и единиц, представляющая ее входные данные. Данные отделяются от кодирования машины тремя последовательными c. Первоначально верхняя дорожка будет вся заполнена символами B, за исключением двух ячеек: третьего символа c в цепочке ccc в начале кода машины и первой ячейки данных — см. ниже:

m m

ccc блок сост. 1 cc блок сост. 2 cc ... cc блок последнего состояния $\underbrace{ccc\ 0110...}_{\text{данные}}$

Универсальная машина Тьюринга U будет моделировать движения, которые бы предпринимала Tm T на входной цепочке, представленной как данные на ленте машины U, следующим образом.

Во-первых, машина U передвигает свою головку вправо, пока она не обнаружит маркер в области данных над входным символом, сканируемым машиной T. Этот символ, назовем его A, запоминается в конечном управлении машины U, которая с этого момента начинает движение влево, пока не достигнет маркера, регистрирующего текущее состояние машины T (отмечающего соответствующий блок в коде таблицы машины T). Машина U удаляет этот маркер (заменяет его символом B), передвигается вправо к подблоку, соответствующему символу A, и помещает маркер над первым символом в этом подблоке при условии, что он есть 1. Если же он — 0, то машина U останавливается, поскольку нет никакого следующего движения у машины T. В последующем этот маркер подблока будем называть m_1.

Предположим, что первый символ был 1. Тогда машина U движется влево, пока не находит цепочку ccc. Затем машина U движется вправо, помечая крайнее правое из этих трех c. Этот маркер назовем m_2. Машина U продолжает продвигаться вправо, пока не будет найден маркер m_1. После этого машина U входит в подпрограмму, которая попеременно передвигает маркер m_1 на одну 1 вправо, а маркер m_2 — на один блок вправо. Чтобы отличить маркеры, которые оба есть m, машина U будет запоминать в своем конечном управлении, какой маркер она видела последним. Когда машина U сдвигает m_1 на символ, который не является 1, маркер m_2 располагается над символом c, который как раз перед блоком, соответствующим следующему состоянию машины T. В этой точке машина U удаляет маркер m_1 и записывает в своем конечном управлении символ, который машина T будет печатать, и направление, в котором машина T будет двигать свою головку ленты. Затем машина U снова движется вправо в область данных и находит маркер, который указывает место головки ленты машины T. Символ, находящийся под маркером, заменяется на символ, запомненный в конечном управлении, а маркер сдвигается на одну ячейку в направлении, также зафиксированном в конечном управлении. Таким образом машина U смоделировала одно движение машины T.

Далее машина U запоминает новый символ, сканируемый машиной T, в своем конечном управлении, начинает двигаться влево, пока не достигнет маркера m_2, регистрирующего состояние машины T, и повторяет процесс, который был только что описан.

Если машина T останавливается с этими конкретными данными, то машина U, в точности воспроизведя все движения машины T, тоже останавливается, а в области данных будет зафиксировано финальное содержание ленты машины T.

Когда машина T останавливается, машина U может сообщить, находится ли машина T в принимающем состоянии или нет.

Если машина T не останавливается, то машина U тоже не останавливается, т.е. не принимает.

Так наша универсальная машина Тьюринга моделирует машину Тьюринга T. Детальное устройство универсальной машины Тьюринга, которую мы описали неформально, дано в табл. 7.2.

Отметим, что эта универсальная машина Тьюринга имеет 12 ленточных символов, но может моделировать только машину Тьюринга с двумя допустимыми символами ленты. Но можно построить эквивалентную универсальную машину Тьюринга, которая будет использовать только два допустимых символа ленты. Для этого каждый допустимый символ ленты надо закодировать блоком из четырех символов 0 или 1. Часть первоначальной ленты с данными будет использовать четыре ячейки вместо одной непустой ячейки на оригинальной входной ленте машины T.

§ 7.2. Неразрешимость проблемы остановки

Проблема остановки машины Тьюринга формулируется следующим образом: дана машина Тьюринга в произвольной конфигурации со строкой непустых ленточных символов конечной длины. Остановиться ли она в конце концов?

Говорят, что эта проблема рекурсивно не разрешима в том смысле, что не существует алгоритма, который для каждой Tm и каждой конфигурации определял бы, остановится ли машина когда-нибудь. Это совсем не значит, что мы не можем определить, остановится ли конкретная Tm в конкретной ситуации.

При описании универсальной машины Тьюринга мы имели кодирование для любой Tm с ленточными символами 0, 1 и B. Кодированием была цепочка из $\{0, 1, c, L, R\}^*$. Мы можем перенумеровать все такие цепочки, перечисляя их в порядке возрастания длины. Цепочки одинаковой длины упорядочиваются в соответствии со значением строки по основанию 5. Предполагается, что эти 5 символов играют роль целых 0, 1, 2, 3, 4 в каком-нибудь порядке. Аналогичным образом цепочки из множества $\{0, 1\}^*$ могут быть тоже упорядочены. Первыми цепочками являются ε, 0, 1, 00, 01, 10, 11, 000, 001, ... Таким образом, имеет смысл говорить об i-й цепочке в множестве $\{0, 1\}^*$.

Если мы предположим, что каждая цепочка из множества $\{0, 1, c, L, R\}^*$ является машиной Тьюринга (некоторые цепочки будут образованы неправильно — они рассматриваются как Tm без каких-нибудь движений), то также имеет смысл говорить о j-й машине Тьюринга, т.е. о машине, представленной j-й цепочкой из множества $\{0, 1, c, L, R\}^*$.

Рассмотрим язык $L_1 = \{x_i \mid x_i$ не принимается Tm $T_i\}$. Ясно, что язык L_1 не мог бы приниматься никакой Tm. Если бы это не было так, то существовала бы некоторая машина Тьюринга, скажем T_j, которая бы принимала язык L_1. Возьмем, цепочку x_j. Если $x_j \in L_1$, то по определению языка x_j не принимается машиной T_j. С другой стороны, машина T_j распознает язык L_1, стало быть принимает $x_j \in L_1$. Наше допущение привело к противоречию. Если $x_j \notin L_1$, то x_j не принимается машиной T_j, но тогда по определению языка x_j принимается машиной T_j. Опять противоречие. Остается признать, что язык L_1 не принимается никакой Tm.

Предположим, что мы имели бы алгоритм (т.е. машину Тьюринга, которая всегда останавливается) для определения, остановится ли когда-нибудь машина Тьюринга в данной конфигурации. Обозначим этот алгоритм T_0. Тогда мы могли бы построить машину Тьюринга T, которая принимает язык L_1, а это противоречило бы только что установленному факту. Машина T действовала бы следующим образом:

1. Пусть $x \in L_1$ на ее входе. Прежде всего она перечисляет предложения x_1, x_2, ... до тех пор, пока не обнаружит, что некоторое $x_i = x$. Таким образом Tm T определяет, что x является i-м предложением в этом перечислении.

2. Затем Tm T генерирует код машины Тьюринга T_i.

3. Управление теперь передается машине T_0, которая может определить, останавливается Tm T_i с входной цепочкой x_i или нет.

4. Если установлено, что Tm T_i не останавливается с входной цепочкой x_i (т.е. Tm T_i не принимает x_i), то машина T останавливается и принимает.

5. Если же установлено, что Tm T_i останавливается с входной цепочкой x_i, то управление передается универсальной машине Тьюринга, которая моделирует машину T_i с входной цепочкой x_i.

6. Поскольку, как было выяснено на предыдущем шаге, Tm T_i останавливается с входной цепочкой x_i, то универсальная машина Тьюринга остановится и определит, принимает машина T_i цепочку x_i или нет. В любом случае Tm T останавливается, принимая x_i в случае, когда Tm T_i цепочку x_i не принимает, и наоборот, отвергая ее, если Tm T_i ее принимает.

Итак, наше предположение о том, что существует машина Тьюринга, которая всегда останавливается и решает проблему остановки произвольной машины Тьюринга, привела нас к противоречию, состоящему в том, что мы сумели построить машину Тьюринга, распознающую язык L_1. Это дает возможность сформулировать следующее утверждение.

Теорема 7.1. *Не существует алгоритма (машины Тьюринга, которая гарантированно останавливается) для определения, остановится ли в конце концов произвольная машина Тьюринга, начиная в произвольно заданной конфигурации.*

Доказательство вытекает из подходящей формализации вышеприведенного рассуждения.

Для многих проблем не существует разрешающего алгоритма, в частности и для решения некоторых проблем, касающихся теории языков.

§ 7.3. Класс рекурсивных множеств

Мы можем теперь показать, что класс рекурсивных множеств является собственным подмножеством рекурсивно перечислимых множеств. Другими словами, существует множество, предложения которого могут быть распознаны машиной Тьюринга, которая не останавливается на некоторых предложениях не из этого множества, но не могут быть распознаны никакой машиной Тьюринга, которая всегда останавливается.

Примером такого множества является дополнение множества L_1, о котором шла речь в предыдущем параграфе. Прежде, чем доказать это, дадим две леммы.

Лемма 7.1. *Если множество рекурсивно, то его дополнение рекурсивно.*

Доказательство. Если $L \subseteq \Sigma^*$ — рекурсивное множество, то существует машина Тьюринга T, гарантированно останавливающаяся, которая принимает язык L. Можно предполагать, что после принятия входной цепочки Tm T не делает больше никаких движений. Построим другую машину Тьюринга — T_1, у которой одно принимающее состояние: q. Правила Tm T_1 включают все правила машины T, так что Tm T_1 моделирует Tm T. Кроме того, функция δ Tm T_1 доопределяется для непринимающих состояний и допустимых символов ленты, для которых дальнейшее движение не определено, движением, переводящим Tm T_1 в принимающее состояние q. В состоянии q машина T_1 останавливается, принимая входную цепочку, которая первоначальной машиной T не принимается.

Так Tm T_1 моделирует Tm T до тех пор, пока Tm T не останавливается. Если машина T останавливается в непринимающем состоянии, она, конечно, не принимает свою входную цепочку, но Tm T_1 делает еще одно движение в состояние q и принимает. Ясно, что Tm T_1 принимает язык $\Sigma^* \setminus L$. Что и требовалось доказать.

Лемма 7.2. *Пусть $x_1, x_2, \ldots$ — эффективное перечисление всех предложений над некоторым конечным алфавитом Σ, а $T_1, T_2, \ldots$ — эффективное перечисление всех машин Тьюринга с символами ленты, выбранными из некоторого конечного алфавита, включающего Σ. Пусть $L_2 = \{ x_i \mid x_i$ принимается машиной $T_i \}$.*

Утверждается, что L_2 — рекурсивно перечислимое множество, дополнение которого не является рекурсивно перечислимым.

Доказательство. Предложения языка L_2 могут приниматься машиной Тьюринга T, которая не обязательно останавливается на предложениях, не принадлежащих языку L_2, и действует следующим образом.

Для данного предложения x машина T перечисляет цепочки x_1, x_2, ... до тех пор, пока она не находит цепочку $x_i = x$, тем самым определяя, что x является i-й цепочкой в перечислении.

Затем Tm T генерирует Tm T_i и передает управление универсальной машине Тьюринга, которая моделирует Tm T_i с входной цепочкой x_i.

Если Tm T_i с входой цепочкой x_i останавливается и принимает, то Tm T тоже останавливается, принимая. Если Tm T_i останавливается и отвергает x_i, то Tm T тоже останавливается, отвергая. Наконец, если Tm T_i не останавливается, то Tm T тоже не останавливается.

Таким образом, множество L_2 — рекурсивно перечислимо, поскольку оно принимается Tm T.

Кроме того, множество $\overline{L_2}$ не может быть рекурсивно перечислимым, так как если T_j — машина Тьюринга, принимающая множество $\overline{L_2}$, то предложение x_j принадлежит множеству $\overline{L_2}$ тогда и только тогда, когда предложение x_j не принимается Tm T_j. Это противоречит утверждению, что $\overline{L_2}$ — язык, принимаемый Tm T_j. Что и требовалось доказать.

Теорема 7.2. Существует рекурсивно перечислимое множество, которое не является рекурсивным.

Доказательство. Согласно лемме 7.2 L_2 — рекурсивно перечислимое множество, дополнение которого не является рекурсивно перечислимым. Теперь, если бы L_2 было рекурсивным, то по лемме 7.1 его дополнение, $\overline{L_2}$, тоже было бы рекурсивным и, следовательно, рекурсивно перечислимым, что противоречило бы утверждению леммы 7.2. Что и требовалось доказать.

§ 7.4. Машины Тьюринга и грамматики типа 0

В этом параграфе мы докажем, что язык распознается машиной Тьюринга тогда и только тогда, когда он порождается грамматикой типа 0.

Чтобы доказать *достаточность*, мы построим недетерминированную машину Тьюринга, которая недетерминированно выбирает вывод в грамматике и смотрит, совпадает ли результат этого вывода с входной цепочкой. Если да, то машина принимает ее.

Чтобы доказать *необходимость*, мы строим грамматику, которая порождает представление терминальной строки, а затем моделирует машину Тьюринга на этой строке. Если строка принимается машиной, то строка преобразуется к терминальным символам, которые она представляет.

Теорема 7.3. *Если язык L порождается грамматикой типа* 0, *то язык L распознается машиной Тьюринга*.

Доказательство. Пусть $G = (V_N, V_T, P, S)$ — грамматика типа 0 и $L = L(G)$. Опишем неформально машину Тьюринга T, принимающую язык L. Машина T будет недетерминированной.

Пусть $T = (Q, V_T, \Gamma, \delta, q_0, F)$, где $\Gamma = V_N \cup V_T \cup \{B, \#, X\}$, причем $B, \#, X \notin V_N \cup V_T$. Мы не перечисляем всех состояний во множестве Q, но назначаем некоторые из них, как только в них возникает потребность. Мы разрешаем Tm T печатать пробел B, если необходимо.

Сначала Tm T имеет ввод $w \in V_T^*$ на ее ленте. Затем, сдвигая цепочку w на одну ячейку вправо, вставляет на освободившееся перед ней место символ #. Следом за w печатается цепочка $\#S\#$. Содержание ленты в этот момент имеет вид $\#w\#S\#$. С этого момента Tm T будет недетерминированно моделировать вывод в грамматике G, начиная с символа S. Каждая сентенциальная форма в выводе будет появляться по очереди между двумя последними ограничителями #. Если некоторый выбор движений приводит к цепочке терминалов, то она сравнивается с w. Если эти две цепочки равны, то Tm T принимает.

Формально пусть в какой-то момент Tm T имеет на своей ленте цепочку вида $\#w\#A_1A_2...A_k\#$. Машина T передвигает свою головку по цепочке $A_1A_2...A_k$, недетерминированно выбирая позицию i и константу r между 1 и максимальной длиной левой части любого правила из множества P.

Затем Tm T исследует подцепочку $A_iA_{i+1}...A_{i+r-1}$. Если она является левой частью некоторого правила из множества P, то ее можно заменить правой частью этого же правила. Машина T может быть вынуждена сдвигать $A_{i+r}A_{i+r+1}...A_k\#$ влево или вправо, чтобы освободить место или заполнить пространство, если длина правой части не равна r. При сдвиге вправо символ X используется для временного заполнения освободившегося пространства.

Из этого простого моделирования выводов в грамматике G должно быть ясно, что Tm T будет печатать на своей ленте строку вида $\#w\#\alpha\#$, где $\alpha \in V^*$, точно тогда, когда $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$. Кроме того, если $\alpha = w$, то Tm T принимает. Заметим, что для реализации проверки этого равенства опять пригодится символ X. Что и требовалось доказать.

Теорема 7.4. *Если язык L распознается машиной Тьюринга*, *то язык L порождается грамматикой типа* 0.

Доказательство. Пусть язык L принимается машиной Тьюринга $T = (Q, \Sigma, \Gamma, \delta, q_0, F)$. Мы построим грамматику G, которая недетерминированно порождает две копии представления некоторого слова из множества Σ^*, а затем моделирует действие Tm T на одной из этих копий. Если Tm T принимает слово, то грамматика G превращает вторую копию в терминальную строку. Если Tm T не принимает слово, вывод никогда не дает в результате терминальную строку.

Снова мы предполагаем без потери общности рассуждений, что для каждого $q \in F$ и $a \in \Sigma$ значение $\delta(q, a)$ не определено.

Формально пусть $G = (V_N, \Sigma, P, A_1)$, где $V_N = \{[X, Y] \mid X \in \Sigma \cup \{\varepsilon\}, Y \in \Gamma\} \cup Q \cup \{A_1, A_2, A_3\}$, а

$P = \{(1)\ A_1 \to q_0 A_2,$

(2) $A_2 \to [a, a]A_2$ для каждого $a \in \Sigma$,

(3) $A_2 \to A_3$,

(4) $A_3 \to [\varepsilon, B]A_3$,

(5) $A_3 \to \varepsilon$,

(6) $q[a, C] \to [a, D]p$ для каждых $a \in \Sigma \cup \{\varepsilon\}$, $q \in Q$, $C \in \Gamma$, таких, что $\delta(q, C) = (p, D, R)$,

(7) $[b, E]q[a, C] \to p[b, E][a, D]$ для всех $C, D, E \in \Gamma$; $a, b \in \Sigma \cup \{\varepsilon\}$, $q \in Q$, таких, что $\delta(q, C) = (p, D, L)$,

(8) $[a, C]q \to qaq$, $q[a, C] \to qaq$, $q \to \varepsilon$ для каждого $a \in \Sigma \cup \{\varepsilon\}$, $C \in \Gamma$ и $q \in F\}$.

Используя правила 1 и 2, получаем вывод вида

$A_1 \overset{*}{\underset{G}{\Rightarrow}} q_0[a_1, a_1]\ [a_2, a_2]\ \ldots\ [a_k, a_k]A_2$, где $a_i \in \Sigma$, $i = 1, 2, \ldots, k$.

Предположим, что Tm T принимает цепочку $a_1 a_2 \ldots a_k$, используя не более, чем m ячеек справа от своего ввода. Тогда, используя правило 3, затем m раз правило 4 и, наконец, правило 5, продолжим предыдущий вывод. Получим

$$A_1 \overset{*}{\underset{G}{\Rightarrow}} q_0[a_1, a_1]\ [a_2, a_2]\ \ldots\ [a_k, a_k]\ [\varepsilon, B]^m.$$

Заметим, что с этого момента и впредь только правила 6 и 7 могут использоваться до тех пор, пока не порождается принимающее состояние. При этом первые компоненты в обозначениях нетерминалов никогда не изменяются, а вторые моделируют записи, производимые Tm T на ее ленте.

Индукцией по числу l движений машины T можно показать, что если $(q_0, a_1 a_2 \ldots a_k, 1) \overset{*}{\underset{T}{\vdash}} (q, X_1 X_2 \ldots X_s, r)$, то

$q_0[a_1, a_1][a_2, a_2]\ldots[a_k, a_k][\varepsilon, B]^m \overset{*}{\underset{G}{\Rightarrow}} [a_1, X_1][a_2, X_2]\ldots[a_{r-1}, X_{r-1}]q[a_r, X_r]\ldots[a_{k+m}, X_{k+m}]$,

где $a_1, a_2, \ldots, a_k \in \Sigma$; $a_{k+1} = a_{k+2} = \ldots = a_{k+m} = \varepsilon$; $X_1, X_2, \ldots, X_{k+m} \in \Gamma$; $X_{s+1} = X_{s+2} = \ldots = X_{k+m} = B$.

База. Пусть $l = 0$. Утверждение выполняется очевидным образом.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \le n$ $(n \ge 0)$.

Индукционный переход. Пусть Tm T выполняет следующие $n+1$ движений:

$$(q_0, a_1 a_2 \ldots a_k, 1) \overset{n}{\underset{T}{\vdash}} (q_1, X_1 X_2 \ldots X_r, j_1) \underset{T}{\vdash} (q_2, Y_1 Y_2 \ldots Y_s, j_2).$$

Тогда по индукционной гипотезе существует вывод вида

$$q_0[a_1, a_1][a_2, a_2]\ldots[a_k, a_k][\varepsilon, B]^m \overset{*}{\underset{G}{\Rightarrow}} [a_1, X_1][a_2, X_2]\ldots q_1[a_{j_1}, X_{j_1}]\ldots[a_{k+m}, X_{k+m}].$$

Судя по последнему движению, должно быть $\delta(q_1, X_{j_1}) = (q_2, Y_{j_1}, D)$ и $D = L$, если $j_2 = j_1 - 1$ или $D = R$, если $j_2 = j_1 + 1$. Соответственно при $D = R$ существует правило грамматики вида 6: $q_1[a_{j_1}, X_{j_1}] \to [a_{j_1}, Y_{j_1}]q_2$; при $D = L$ существует правило грамматики вида 7: $[a_{j_1-1}, X_{j_1-1}]q_1[a_{j_1}, X_{j_1}] \to q_2[a_{j_1-1}, X_{j_1-1}][a_{j_1}, Y_{j_1}]$.

Таким образом, еще один шаг вывода дает

$$[a_1, X_1][a_2, X_2]...q_1[a_{j_1}, X_{j_1}]...[a_{k+m}, X_{k+m}] \underset{G}{\Rightarrow} [a_1, Y_1][a_2, Y_2]...q_2[a_{j_2}, Y_{j_2}]...[a_{k+m}, Y_{k+m}],$$

где $X_i = Y_i$ для всех $i \neq j_1$. Итак, вспомогательное утверждение доказано.

Далее, если $q \in F$, то по правилам грамматики вида 8 можно получить вывод

$$[a_1, X_1][a_2, X_2]... q [a_j, X_j]...[a_{k+m}, X_{k+m}] \overset{*}{\underset{G}{\Rightarrow}} a_1a_2...a_k.$$

Итак, доказано, что если $a_1a_2...a_k$ принимается Tm T, то $a_1a_2...a_k \in L(G)$.

Чтобы завершить доказательство теоремы, остается показать, что если $A_1 \overset{*}{\underset{G}{\Rightarrow}} w$, то цепочка w принимается Tm T. Прежде всего отметим, что любой вывод и, в частности, вывод $A_1 \overset{*}{\underset{G}{\Rightarrow}} w$ может начинаться только по правилам 1–5, дающим результат вида

$$A_1 \overset{*}{\underset{G}{\Rightarrow}} q_0[a_1, a_1][a_2, a_2]...[a_k, a_k][\varepsilon, B]^m.$$

Далее могут применяться правила 6 и 7, дающие в конце концов результат вида

$$[a_1, X_1][a_2, X_2]...q[a_j, X_j]...[a_{k+m}, X_{k+m}],$$

где $q \in F$, после чего правила 8 дадут $w = a_1a_2...a_k$.

Собственно, надо показать, что движения Tm T моделируются на участке вывода

$$q_0[a_1, a_1][a_2, a_2]...[a_k, a_k][\varepsilon, B]^m \overset{*}{\underset{G}{\Rightarrow}} [a_1, Y_1][a_2, Y_2]...q[a_j, Y_j]...[a_{k+m}, Y_{k+m}],$$

где $q \in F$. Другими словами, если такой вывод имеет место, то существуют движения Tm T вида $(q_0, a_1a_2...a_k, 1) \overset{*}{\underset{T}{\vdash}} (q, Y_1Y_2...Y_{k+m}, j)$. Докажем это утверждение индукцией по l — длине вывода.

База. Пусть $l = 0$. Утверждение выполняется очевидным образом.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \leq n$ $(n \geq 0)$.

Индукционный переход. Пусть имеется вывод длиной $l = n + 1$. В общем случае имеем

$$q_0[a_1, a_1][a_2, a_2]...[a_k, a_k][\varepsilon, B]^m \overset{n}{\underset{G}{\Rightarrow}}$$
$$\overset{n}{\underset{G}{\Rightarrow}} [a_1, X_1][a_2, X_2]...q_1[a_{j_1}, X_{j_1}]...[a_{k+m}, X_{k+m}] \underset{G}{\Rightarrow}$$
$$\underset{G}{\Rightarrow} [a_1, Y_1][a_2, Y_2]...q[a_j, Y_j]...[a_{k+m}, Y_{k+m}],$$

где $q \in F$. Согласно индукционной гипотезе существует переход

$$(q_0, a_1a_2...a_k, 1) \overset{*}{\underset{T}{\vdash}} (q_1, X_1X_2...X_{k+m}, j_1).$$

Ясно, что последний шаг вывода мог быть выполнен только посредством правила вида 6 или 7. Если применялось правило вида 6, то $j = j_1 + 1$; если

использовалось правило вида 7, то $j = j_1 - 1$. Эти правила существуют благодаря тому, что $\delta(q_1, X_{j_1}) = (q, Y_{j_1}, D)$, где $D = R$, если $j = j_1 + 1$, или $D = L$, если $j = j_1 - 1$. При этом $X_i = Y_i$ для всех $i \neq j_1$. Благодаря этим значениям функции δ машина T совершает еще одно движение, переводящее ее в принимающую конфигурацию:

$$(q_0, a_1a_2...a_k, 1) \overset{*}{\underset{T}{\vdash}} (q_1, X_1X_2...X_{k+m}, j_1) \underset{T}{\vdash} (q, Y_1Y_2...Y_{k+m}, j),$$

где $q \in F$. Другими словами, показано, что $w = a_1a_2...a_k$ принимается Tm T.

Итак, если $w \in L(G)$, то цепочка w принимается машиной T. Теорема доказана полностью.

Глава 8

ЛИНЕЙНО ОГРАНИЧЕННЫЕ АВТОМАТЫ И КОНТЕКСТНО-ЗАВИСИМЫЕ ЯЗЫКИ

§ 8.1. Линейно ограниченные автоматы

Линейно ограниченный автомат (lba) есть недетерминированная одноленточная машина Тьюринга, которая никогда не покидает те ячейки, на которых размещен ее ввод. Формально он определяется следующим образом.

***Определение* 8.1.** *Линейно ограниченным автоматом* называется формальная система $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, в которой Q — *множество состояний*; $q_0 \in Q$ — *начальное состояние*; $F \subseteq Q$ — *множество конечных состояний*; Γ — *алфавит допустимых символов ленты*; $\Sigma \subseteq \Gamma$ — *алфавит входных символов*, который содержит два особых символа: ¢ и \$ — *левый* и *правый маркеры*, находящиеся с самого начала на концах входной цепочки для того, чтобы предотвращать выход головки ленты за пределы участка, на котором размещается входная цепочка (считается, что маркеры могут использоваться только в этой роли: на место маркера нельзя записать какой-нибудь другой символ ленты, и никакой символ ленты не может быть заменен каким-нибудь маркером); δ — отображение типа $Q \times \Gamma \to 2^{Q \times \Gamma \times \{L, R\}}$.

Движения линейно ограниченного автомата описываются в терминах *конфигураций*. Конфигурация lba M обозначается как $(q, A_1A_2\ldots A_n, i)$, где $q \in Q$; $A_1, A_2, \ldots, A_n \in \Gamma$; i — целое, причем $1 \le i \le n$. Согласно определению $A_1 = ¢$, $A_n = \$$.

Если $(p, A, L) \in \delta(q, A_i)$ и $i > 1$, то

$$(q, A_1A_2\ldots A_n, i) \vdash_M (p, A_1A_2\ldots A_{i-1}AA_{i+1}\ldots A_n, i-1).$$

Если $(p, A, R) \in \delta(q, A_i)$ и $i < n$, то

$$(q, A_1A_2\ldots A_n, i) \vdash_M (p, A_1A_2\ldots A_{i-1}AA_{i+1}\ldots A_n, i+1).$$

Другими словами, lba M печатает символ A на месте A_i, изменяет свое состояние на p и продвигает свою головку влево или вправо, не выходя из области, в которой символы находились изначально.

Отношения на множестве конфигураций $\vdash_M$, $\vdash_M^n$, $\vdash_M^+$ или $\vdash_M^*$ определяются обычным для недетерминированных машин Тьюринга образом.

***Определение* 8.2.** *Языком, принимаемым линейно ограниченным автоматом* M, называется множество

$$\{w \mid w \in (\Sigma \setminus \{¢, \$\})^*, (q_0, ¢w\$, 1) \vdash_M^* (q, ¢\alpha\$, i), q \in F, \alpha \in \Gamma^*, 1 \le i \le n, n = |w| + 2\}.$$

О*пределение* 8.3. *Линейно ограниченный автомат* M является *детерминированным*, если $\#\delta(q, A) \le 1$ для любых $q \in Q$, $A \in \Gamma$.

Неизвестно, является ли класс множеств, распознаваемых детерминированными линейно ограниченными автоматами, строгим подклассом множеств, распознаваемых недетерминированными lba, или они совпадают. Конечно, справедливо, что любое множество, принимаемое недетерминированным линейно ограниченным автоматом, принимается некоторой детерминированной машиной Тьюринга. Однако длина ленты, требуемая этой Tm, может быть экспоненциальной, а не линейной, функцией от длины входной цепочки.

§ 8.2. Связь линейно ограниченных автоматов с контекстно-зависимыми языками

Наш интерес к недетерминированным линейно ограниченным автоматам проистекает из того факта, что класс множеств, принимаемых ими, является в точности классом контекстно-зависимых языков.

Теорема 8.1. *Если L — контекстно-зависимый язык, то язык L принимается некоторым линейно ограниченным автоматом.*

Доказательство. Пусть $G = (V_N, V_T, P, S)$ — контекстно-зависимая грамматика. Мы построим lba M, такой, что язык, принимаемый lba M, есть $L(G)$. Мы не будем вдаваться в детальное построение автомата M, поскольку оно очень сложно, а просто опишем, как он работает.

Входная лента будет иметь две дорожки. Дорожка 1 будет содержать входную строку x ($x \neq \varepsilon$) с концевыми маркерами. Дорожка 2 будет использоваться для работы.

На первом шаге lba M помещает символ S в крайнюю левую ячейку дорожки 2. Затем автомат входит в порождающую подпрограмму, которая выполняет следующие шаги:

1. Подпрограмма выбирает последовательные подстроки символов α на дорожке 2, такие, что $\alpha \to \beta \in P$.

2. Подстроки α заменяются на β, сдвигая вправо, если необходимо, символы, расположенные справа от α. Если эта операция заставляет символ быть вытолкнутым за правый маркер, автомат останавливается. Как известно, промежуточные сентенциальные формы в контекстно-зависимой грамматике не длиннее, чем выводимая терминальная цепочка. Так что, если на очередном шаге получена сентенциальная форма длиннее x, то продолжать процесс не имеет смысла, потому что все последующие сентенциальные формы будут разве лишь длиннее.

3. Подпрограмма недетерминированно выбирает, возвращаться ли к шагу 1, либо идти на выход.

4. При выходе из подпрограммы дорожка 1 все еще будет содержать строку x, в то время как дорожка 2 будет содержать некоторую строку γ, такую, что $S \overset{*}{\underset{G}{\Rightarrow}} \gamma$. Автомат M сравнивает посимвольно цепочки x и γ. Если окажется, что $x \neq \gamma$, то автомат останавливается, не принимая; если же окажется, что $x = \gamma$, то он останавливается, принимая входную цепочку.

Ясно, что если $x \in L(G)$, то найдется такая последовательность движений lba M, которая сгенерирует цепочку x на дорожке 2, и тогда автомат остановится, принимая. Аналогично, если lba M принимает цепочку x, то должна существовать последовательность движений, генерирующих цепочку x на дорожке 2. Только при таком условии lba M принимает цепочку x. Но, по построению, процесс генерации x воспроизводит вывод *этой цепочки* из S. Следовательно, $S \overset{*}{\underset{G}{\Rightarrow}} x$. Теорема доказана.

Теорема 8.2. *Если язык L принимается линейно ограниченным автоматом, то L — контекстно-зависимый язык.*

Доказательство. Построение контекстно-зависимой грамматики, которая моделирует линейно ограниченный автомат, подобно построению, описанному в теореме 7.4, в которой грамматика типа 0 строилась, чтобы моделировать движения машины Тьюринга. Нетерминалы контекстно-зависимой грамматики должны указывать не только первоначальное содержание некоторой ячейки ленты lba, но также и то, является ли эта ячейка смежной с концевым маркером

слева или справа. Такие ячейки в обозначении нетерминалов мы будем снабжать маркерами $\not\in$ и \$, обозначающими, что ячейка граничит соответственно с левым, правым или обоими концевыми маркерами. В обозначении нетерминала состояние lba должно также комбинироваться с символом, находящимся под головкой ленты. Контекстно-зависимая грамматика не может иметь отдельных символов для концевых маркеров и состояния линейно ограниченного автомата, потому что эти символы должны были бы заменяться на пустые цепочки, когда строка превращается в терминальную, а ε-порождения в csg запрещены.

Формально пусть $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ — недетерминированный lba, где $\not\in, \$ \not\in \Sigma$ и L — язык, принимаемый lba M, причем $\varepsilon \not\in L$.

Положим $G = (V_N, V_T, P, A_1)$, где

$V_N = \{A_1, A_2\} \cup \{[q, \not\in, X, a, \$], [\not\in, q, X, a, \$], [\not\in, X, a, q, \$], [q, X, a], [q, X, a, \$],$
$[X, a, q, \$], [\not\in, X, a], [X, a], [X, a, \$] \mid a \in \Sigma \setminus \{\not\in, \$\}, X \in \Gamma \setminus \{\not\in, \$\}, q \in Q\}$;

$V_T = \Sigma$; а множество P включает следующие правила:

(1) $A_1 \to [q_0, \not\in, a, a, \$]$ — моделируют начальные конфигурации вида $(q_0, \not\in a\$, 1)$;

(2.1) $[q, \not\in, X, a, \$] \to [\not\in, p, X, a, \$]$, если $(p, \not\in, R) \in \delta(q, \not\in)$;
(2.2) $[\not\in, q, X, a, \$] \to [p, \not\in, Y, a, \$]$, если $(p, Y, L) \in \delta(q, X)$;
(2.3) $[\not\in, q, X, a, \$] \to [\not\in, Y, a, p, \$]$, если $(p, Y, R) \in \delta(q, X)$;
(2.4) $[\not\in, X, a, q, \$] \to [\not\in, p, X, a, \$]$, если $(p, \$, L) \in \delta(q, \$)$;

} Моделируют движения на односимвольной цепочке при $q \in Q \setminus F$.

(3.1) $[q, \not\in, X, a, \$] \to a$;
(3.2) $[\not\in, q, X, a, \$] \to a$;
(3.3) $[\not\in, X, a, q, \$] \to a$;

} Восстанавливают входную односимвольную цепочку при $q \in F$.

(4.1) $A_1 \to [q_0, \not\in, a, a] A_2$;
(4.2) $A_2 \to [a, a] A_2$;
(4.3) $A_2 \to [a, a, \$]$;

} Моделируют начальные конфигурации вида $(q_0, \not\in w\$, 1)$, где $|w| > 1$.

(5.1) $[q, \not\in, X, a] \to [\not\in, p, X, a]$, если $(p, \not\in, R) \in \delta(q, \not\in)$;
(5.2) $[\not\in, q, X, a] \to [p, \not\in, Y, a]$, если $(p, Y, L) \in \delta(q, X)$;
(5.3) $[\not\in, q, X, a]\,[Z, b] \to [\not\in, Y, a]\,[p, Z, b]$, если $(p, Y, R) \in \delta(q, X)$;

} Моделируют движения на левом конце цепочки при $q \in Q \setminus F$.

(6.1) $[q, X, a]\,[Z, b] \to [Y, a][p, Z, b]$, если $(p, Y, R) \in \delta(q, X)$;
(6.2) $[Z, b]\,[q, X, a] \to [p, Z, b]\,[Y, a]$, если $(p, Y, L) \in \delta(q, X)$;
(6.3) $[q, X, a]\,[Z, b, \$] \to [Y, a]\,[p, Z, b, \$]$, если $(p, Y, R) \in \delta(q, X)$;

} Моделируют движения в середине цепочки при $q \in Q \setminus F$.

(7.1) $[q, X, a, \$] \to [Y, a, p, \$]$, если $(p, Y, R) \in \delta(q, X)$;
(7.2) $[X, a, q, \$] \to [p, X, a, \$]$, если $(p, \$, L) \in \delta(q, \$)$;
(7.3) $[Z, b]\,[q, X, a, \$] \to [p, Z, b]\,[Y, a, \$]$, если $(p, Y, L) \in \delta(q, X)$;

} Моделируют движения на правом конце цепочки при $q \in Q \setminus F$.

(8.1) $[q, \not\in, X, a] \to a$;
(8.2) $[\not\in, q, X, a] \to a$;

} Восстановление входного символа при переходе в состояние $q \in F$ на левом конце цепочки.

(8.3) $[q, X, a] \to a$;

} Восстановление входного символа при переходе в состояние $q \in F$ в середине цепочки.

(8.4) $[q, X, a, \$] \to a$;
(8.5) $[X, a, q, \$] \to a$;

} Восстановление входного символа при переходе в состояние $q \in F$ на правом конце цепочки.

(9.1) $a[X, b] \to ab$;
(9.2) $a[X, b, \$] \to ab$;

} Восстановление входной цепочки слева направо.

(9.3) $[X, a]b \to ab$;
(9.4) $[\not\in, X, a]b \to ab$;

} Восстановление входной цепочки справа налево.

Здесь $p \in Q$; $X, Y, Z \in \Gamma \setminus \{\not\in, \$\}$; $a, b \in \Sigma \setminus \{\not\in, \$\}$.

Очевидно, что построенная нами грамматика G — контекстно-зависима.

Индукцией по числу движений lba M нетрудно доказать, что если непустая цепочка принимается линейно ограниченным автоматом M, то она выводима в контекстно-зависимой грамматике G. И наоборот, индукцией по длине вывода можно доказать, что если цепочка выводима в контекстно-зависимой грамматике G, то она принимается линейно ограниченным автоматом M.

§ 8.3. Контекстно-зависимые языки — подкласс рекурсивных множеств

В гл. 2 было показано, что каждый контекстно-зависимый язык рекурсивен. Теперь мы покажем, что обратное утверждение неверно.

Теорема 8.3. *Существуют рекурсивные множества, которые не являются контекстно-зависимыми*.

Доказательство. Мы можем перечислять все строки из множества $\{0,1\}^*$, как в § 7.3. Пусть x_i — i-я строка. Аналогично мы можем перечислять все грамматики типа 0, у которых алфавит терминалов состоит из тех же символов 0 и 1. Поскольку имена нетерминалов не существенны, а каждая грамматика имеет конечное их число, можно предположить, что имеется не более чем счетное число нетерминалов, которые можно представить в бинарном коде как 01, 011, 0111, 01111, и т.д. Предполагается, что 01 — всегда начальный нетерминал. Кроме того, терминал 0 будем представлять как 00, а терминал 1 — при помощи кода 001. Символ $\rightarrow$ представим кодом 0011, а запятую, отделяющую одно правило грамматики от другого, — кодом 00111. Таким образом, символы алфавитов, используемые в правилах грамматики, представляются двоичными кодами: 00 — терминал 0, 001 — терминал 1 и 01^i, где $i = 1, 2, \ldots$ — нетерминалы. Состав нетерминального словаря конкретной грамматики определяется по самим правилам грамматики, в которых эти нетерминалы используются.

Заметим, что не все цепочки из 0 и 1 представляют грамматики, тем более контекстно-зависимые грамматики. Однако по заданной цепочке можно легко судить, является ли она csg. Поскольку имеется бесконечно много csg, мы можем их нумеровать некоторым разумным образом: $G_1, G_2, \ldots$.

Доказательство теоремы теперь тривиально.

Определим язык $L = \{x_i \mid x_i \notin L(G_i)\}$. Язык L — рекурсивен. Действительно, по любой данной цепочке x_i легко определить i, а затем можно сгенерировать G_i. Согласно теореме 2.2 имеется алгоритм, который определяет, находится ли цепочка x_i в языке $L(G_i)$, поскольку G_i — контекстно-зависимая грамматика. Следовательно, имеем алгоритм определения для любой цепочки x, находится ли она в языке L.

Теперь покажем, что язык L не порождается никакой контекстно-зависимой грамматикой. Предположим противное: язык L порождается csg G_j. Пусть $x_j \in L$. Поскольку $L = L(G_j)$, то $x_j \in L(G_j)$. Но по самому определению языка $x_j \notin L(G_j)$. Получаем противоречие, которое отвергает наше предположение о том, что существует csg G_j, порождающая язык L. Поскольку приведенное рассуждение справедливо для каждой csg G_j в перечислении и поскольку перечисление содержит каждую csg, мы заключаем, что язык L не является контекстно-зависимым языком. Следовательно, L — рекурсивное множество, которое не является контекстно-зависимым. Что и требовалось доказать.

Глава 9

ОПЕРАЦИИ НАД ЯЗЫКАМИ

§ 9.1. Замкнутость относительно элементарных операций

В этой главе мы применяем операции объединения, конкатенации, обращения, замыкания и т.д. к языкам разных типов. Интересно выяснить, какие операции какие классы языков сохраняют, т.е. отображают языки некоторого класса в тот же самый класс. Есть ряд причин интересоваться этим вопросом. Во-первых, знание, сохраняет операция или нет данный класс языков, помогает характеризовать этот класс. Во-вторых, часто бывает легче узнать, что сложный язык относится к некоторому классу, при помощи того факта, что эта принадлежность является результатом различных операций над другими языками в данном классе, чем путем непосредственного конструирования грамматики для этого языка. В-третьих, знание, полученное из изучения операций над языками, может быть использовано при доказательстве теорем, как это было сделано в гл. 7, где мы показали, что класс рекурсивно перечислимых множеств строго содержит рекурсивные множества, используя при доказательстве тот факт, что рекурсивные множества замкнуты относительно дополнения.

Начнем с рассмотрения операций объединения, конкатенации, замыкания Клини и обращения. Используем следующую лемму о “нормальной форме” для контекстно-зависимых языков и языков типа 0:

Лемма 9.1. *Каждый контекстно-зависимый язык порождается контекстно-зависимой грамматикой, в которой все правила имеют форму либо* $\alpha \to \beta$, *где* α *и* β *— цепочки, состоящие из одних только нетерминалов, либо* $A \to b$, *где* A *— нетерминал, а* b *— терминал. Каждый язык типа* 0 *порождается грамматикой типа* 0, *правила которой имеют указанную форму*.

Доказательство. Пусть $G = (V_N, V_T, P, S)$ — контекстно-зависимая грамматика. Каждому $a \in V_T$ сопоставим новый символ X_a. Рассмотрим грамматику $G_1 = (V_N', V_T, P_1, S)$, где $V_N' = V_N \cup \{X_a \mid a \in V_T\}$. Множество P_1 включает все правила вида $X_a \to a$. Если $\alpha \to \beta \in P$, то $\alpha_1 \to \beta_1 \in P_1$, где цепочки α_1 и β_1 получаются из цепочек α и β путем замещения в них каждого терминала a символом X_a. Доказательство тривиально и мы оставляется его читателю в качестве упражнения. Подобное же доказательство применимо к грамматикам типа 0.

Теорема 9.1 *Классы регулярных*, *контекстно-свободных*, *контекстно-зависимых и рекурсивно перечислимых множеств замкнуты относительно объединения, конкатенации, замыкания и обращения*.

Доказательство. Для класса регулярных множеств доказательство было дано в гл. 3.

Рассмотрим две грамматики: $G_1 = (V_N^{(1)}, V_T^{(1)}, P_1, S_1)$ и $G_2 = (V_N^{(2)}, V_T^{(2)}, P_2, S_2)$, причем обе либо контекстно-свободные, либо контекстно-зависимые, либо типа 0. Без потери общности можно предполагать, что $V_N^{(1)} \cap V_N^{(2)} = \varnothing$.

Кроме того, согласно лемме 9.1 и теореме 4.5 можно считать, что правила грамматик G_1 и G_2 имеют форму $\alpha \to \beta$ и $A \to a$, где α и β — цепочки, состоящие из одних только нетерминалов, A — одиночный нетерминал, а a — одиночный терминальный символ. Кроме того, если G_1 и G_2 — контекстно-свободные грамматики, то $\beta = \varepsilon$ подразумевает, что α есть S_1 или S_2 и что α никогда не появляется в правой части никакого правила.

Объединение. Пусть $G_3 = \left(V_N^{(1)} \cup V_N^{(2)} \cup \{S_3\}, V_T^{(1)} \cup V_T^{(2)}, P_3, S_3\right)$, где $S_3 \notin V_N^{(1)} \cup V_N^{(2)}$, а множество P_3 содержит правила $S_3 \to S_1$, $S_3 \to S_2$ и все правила из множеств P_1 и P_2 за исключением $S_1 \to \varepsilon$ и $S_2 \to \varepsilon$, если G_1 и G_2 — контекстно-зависимы. В случае, когда G_1 и G_2 — контекстно-зависимы и $S_1 \to \varepsilon \in P_1$ или $S_2 \to \varepsilon \in P_2$, добавим правило $S_3 \to \varepsilon$ к множеству правил P_3. Теперь грамматика G_3 — того же типа, что и грамматики G_1, G_2, и $L(G_3) = L(G_1) \cup L(G_2)$.

Конкатенация. Пусть $G_4 = (V_N^{(1)} \cup V_N^{(2)} \cup \{S_4\}, V_T^{(1)} \cup V_T^{(2)}, P_4, S_4)$, где $S_4 \notin V_N^{(1)} \cup V_N^{(2)}$, а множество P_4 содержит правило $S_4 \to S_1 S_2$ и все правила из множеств P_1 и P_2, за исключением правил $S_1 \to \varepsilon$ и $S_2 \to \varepsilon$, если G_1 и G_2 — контекстно-зависимы. В случае, когда G_1 и G_2 — контекстно-зависимы и $S_1 \to \varepsilon \in P_1$, добавим правило $S_4 \to S_2$ к множеству правил P_4; если $S_2 \to \varepsilon \in P_2$, то добавим правило $S_4 \to S_1$ к множеству P_4. Если $S_1 \to \varepsilon \in P_1$ и $S_2 \to \varepsilon \in P_2$, то добавим правило $S_4 \to \varepsilon$ к множеству P_4. Теперь грамматика G_4 — того же типа, что и грамматики G_1, G_2 и $L(G_4) = L(G_1)L(G_2)$.

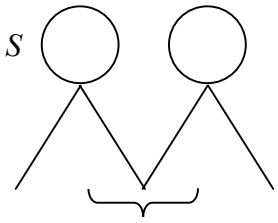

Рис. 9.1.

Заметим, что поскольку $V_N^{(1)} \cap V_N^{(2)} = \varnothing$ и все правила из множеств P_1, P_2 имеют нетерминалы исключительно слева, невозможно, чтобы строка, образованная правым концом сентенциальной формы грамматики G_1, за которой следует левый конец сентенциальной формы грамматики G_2, могла быть левой стороной правила в множестве P_4. Это означает, что левая часть любого правила целиком состоит из нетерминалов только одной из двух грамматик (см. рис. 9.1). Соответственно и все правило относится к одной исходной грамматике. Доказательство того, что $L(G_4) = L(G_1)L(G_2)$ просто.

Замыкание. Пусть $G_5 = (V_N,\ V_T^{(1)},\ P_5,\ S_5)$, где $V_N = V_N^{(1)} \cup \{S_5,\ S_5'\}$, а $P_5 = P_1 \cup \{S_5 \to S_1S_5,\ S_5 \to \varepsilon\}$, если G_5 — контекстно-свободная грамматика, иначе $P_5 = P_1 \cup \{S_5 \to \varepsilon,\ S_5 \to S_1,\ S_5 \to S_1S_5'\} \cup \{aS_5' \to aS_1,\ aS_5' \to aS_1S_5' \mid a \in V_T^{(1)}\}$. Однако в случае, когда G_1 — контекстно-зависимая грамматика, правило $S_1 \to \varepsilon$, если оно имеется, отбрасывается. Грамматика G_5 является грамматикой того же типа, что и G_1, и $L(G_5) = (L(G_1))^*$.

Теперь рассмотрим операции *пересечения* и *дополнения*.

Теорема 9.2. *Класс контекстно-свободных языков не замкнут относительно пересечения*.

Доказательство. Языки $L_1 = \{a^nb^nc^i \mid n \ge 1,\ i \ge 0\}$ и $L_2 = \{a^jb^nc^n \mid n \ge 1,\ i \ge 0\}$ являются контекстно-свободными, поскольку они порождаются грамматиками $G_1 = (\{S, T\}, \{a, b, c\}, \{S \to Sc, S \to T, T \to aTb, T \to ab\}, S)$ и $G_2 = (\{S, T\}, \{a, b, c\}, \{S \to aS, S \to T, T \to bTc, T \to bc\}, S)$ соответственно. Теперь язык $L = L_1 \cap L_2 = \{a^nb^nc^n \mid n \ge 1\}$, который не контекстно-свободен по тривиальному следствию из теоремы 4.7. Действительно, все цепочки L не соответствуют требуемому условию: в L должна быть цепочка $uvwxy$, такая, что все цепочки вида uv^nwx^ny при любом n тоже принадлежали бы языку L. Мы могли бы считать $u = y = \varepsilon$, но ядро w должно быть в первой степени, а в цепочках из языка L оно тоже в степени n.

Теорема 9.3. *Класс контекстно-свободных языков не замкнут относительно дополнения*.

Доказательство. Поскольку класс контекстно-свободных языков замкнут относительно объединения, но не пересечения, то он не может быть замкнут относительно дополнения, поскольку $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Теорема 9.4. *Класс контекстно-свободных языков замкнут относительно пересечения с регулярным множеством*.

Доказательство. Пусть L — контекстно-свободный язык, а R — регулярное множество. Предположим, что $P_1 = (Q_P, \Sigma, \Gamma, \delta_P, p_0, Z_0, F_P)$ — недетерминированный магазинный автомат (npda), принимающий язык L, а $A = (Q_A, \Sigma, \delta_A, q_0, F_A)$ — детерминированный конечный автомат (dfa), принимающий множество R. Построим недетерминированный магазинный автомат (npda) $P_2 = (Q_P \times Q_A, \Sigma, \Gamma, \delta, [p_0, q_0], Z_0, F_P \times F_A)$, который принимает $L \cap R$. Функция δ определяется следующим образом. Для всех $p \in Q_P$, $q \in Q_A$, $a \in \Sigma \cup \{\varepsilon\}$ и $Z \in \Gamma$ функция $\delta([p, q], a, Z)$ содержит $([p', \delta_A(q, a)], \gamma)$ всякий раз, как $\delta_P(p, a, Z)$ содержит (p', γ). (Напомним, что $\delta_A(q, \varepsilon) = q$ для всех $q \in Q_A$.) Неформально npda P_2 хранит след состояний npda P_1 и dfa A в своем конечном управлении.

I. Предположим, что $x \in L \cap R$. Пусть $x = a_1a_2\ldots a_n$, где $a_i \in \Sigma \cup \{\varepsilon\}$, $1 \le i \le n$, так что существуют состояния $q_0, q_1, \ldots, q_n \in Q_A$, $p_0, p_1, \ldots, p_n \in Q_P$ и цепочки $\gamma_0, \gamma_1, \ldots, \gamma_n \in \Gamma^*$, для которых имеют место $\delta_A(q_i, a_{i+1}) = q_{i+1}$ и $(p_i, a_{i+1}\ldots a_n, \gamma_i) \overset{*}{\underset{P_1}{\vdash}}$

$(p_{i+1},\ a_{i+2}\ldots a_n,\ \gamma_{i+1})$ при условии, что $0 \le i < n$, $\gamma_0 = Z_0$, $q_n \in F_A$, $p_n \in F_P$. Тогда $([p_i, q_i], a_{i+1}\ldots a_n, \gamma_i) \overset{*}{\underset{P_2}{\vdash}} ([p_{i+1}, q_{i+1}], a_{i+2}\ldots a_n, \gamma_{i+1})$ и $([p_0, q_0], x, Z_0) \overset{*}{\underset{P_2}{\vdash}} ([p_n, q_n], \varepsilon, \gamma_n)$, при том, что $[p_n, q_n] \in F_P \times F_A$, так что $x \in T(P_2)$.

II. Теперь предположим, что $x \in T(P_2)$. Тогда существуют движения вида $([p_i, q_i], a_{i+1}\ldots a_n, \gamma_i) \overset{*}{\underset{P_2}{\vdash}} ([p_{i+1}, q_{i+1}], a_{i+2}\ldots a_n, \gamma_{i+1})$ для $0 \le i < n$, причем $\gamma_0 = Z_0$, $[p_n, q_n] \in F_P \times F_A$. Тогда $\delta_A(q_i, a_{i+1}) = q_{i+1}$ для $0 \le i < n$, причем $q_n \in F_A$. Следовательно, $x \in R$. Аналогично $(p_i, a_{i+1}\ldots a_n, \gamma_i) \underset{P_1}{\vdash} (p_{i+1}, a_{i+2}\ldots a_n, \gamma_{i+1})$ для $0 \le i < n$ и, как следствие, $(p_0, x, Z_0) \overset{*}{\underset{P_1}{\vdash}} (p_n, \varepsilon, \gamma_n)$. Поскольку $p_n \in F_P$, то $x \in L$.

Из рассуждений I и II следует $T(P_2) = L \cap R$.

Мы уже видели в гл. 3, что класс регулярных множеств относительно пересечения и дополнения. В гл. 7 было показано, что класс рекурсивно перечислимых множеств не замкнут относительно дополнения. Таким образом мы имеем:

Теорема 9.5. *Класс языков типа* 0 *не замкнут относительно дополнения*.

В настоящее время неизвестно, замкнут ли класс контекстно-зависимых языков относительно дополнения. Однако как класс языков типа 0, так и класс контекстно зависимых языков замкнуты относительно пересечения. Доказательства для обоих классов аналогичны, и хотя концептуально просты, утомительны в деталях. Поэтому эти доказательства будут только намечены.

Теорема 9.6. *Класс языков типа* 0 *и класс контекстно-зависимых языков замкнуты относительно пересечения*.

Доказательство. Пусть L_1 и L_2 — языки типа 0 (контекстно-зависимые языки). Рассмотрим две одноленточные машины Тьюринга (два недетерминированных линейно ограниченных автомата) M_1 и M_2, принимающие языки L_1 и L_2 соответственно. Легко построить машину Тьюринга (lba) M, имеющую одну оперативную ленту с тремя дорожками. Первая дорожка содержит ввод. Машина M моделирует машину M_1, используя дорожку 2. Если машина M_1 достигает когда-либо принимающую конфигурацию, то машина M перемещает головку своей ленты на левый конец и моделирует машину M_2 на дорожке 3. Если машина M_2 доходит до принимающей конфигурации, то машина M принимает.

§ 9.2. Замкнутость относительно отображений

Теперь рассмотрим результаты отображений разных типов над языками. Первый тип, который мы рассмотрим, — *подстановка*.

***Определение* 9.1.** *Подстановка* f есть отображение конечного множества Σ на подмножества Δ^* некоторого конечного множества Δ. Другими словами, подстановка f с каждым символом из множества Σ ассоциирует некоторый язык. Отображение f может быть распространено на строки из Σ^* следующим образом:

$f(\varepsilon) = \varepsilon$, $f(xa) = f(x)\,f(a)$, где $x \in \Sigma^*$, $a \in \Sigma$.

Очевидно, что $f(x)$ нужно понимать в обобщенном смысле, тогда как $f(a)$ — в первоначальном.

Мы можем распространить подстановку f далее на языки, определяя $f(L) = \bigcup_{x \in L} f(x)$.

Пример 9.1. Пусть $f(0) = \{a\}$, $f(1) = \{ww^R \mid w \in \{b, c\}^*\}$. Подстановка f отображает множество $\{0^n 1^n \mid n \geq 1\}$ в множество $\{a^n w_1 w_1^R w_2 w_2^R \ldots w_n w_n^R \mid w_i \in \{b, c\}^*$ для $1 \leq i \leq n\}$.

Говорят, что класс языков *замкнут относительно подстановки*, если для любого языка $L \subseteq \Sigma^*$ в данном классе и для любой подстановки f, такой, что $f(a)$ в данном классе для всех $a \in \Sigma$, язык $f(L)$ содержится в этом же классе.

Покажем, что классы регулярных множеств, контекстно-свободных языков и языков типа 0 замкнуты относительно подстановки. Так, в примере 9.1, поскольку $f(0)$ и $f(1)$ — оба контекстно-свободные языки и так как $L = \{0^n 1^n \mid n \geq 1\}$ — контекстно-свободный язык, то множество $f(L) = \{a^n w_1 w_1^R w_2 w_2^R \ldots w_n w_n^R \mid w_i \in \{b, c\}^*$ для $1 \leq i \leq n\}$ также контекстно-свободный язык.

Теорема 9.7. *Классы регулярных множеств, контекстно-свободных языков и языков типа* 0 *замкнуты относительно подстановки*.

Доказательство. Рассмотрим грамматику $G = (V_N, \{a_1, a_2, \ldots, a_n\}, P, S)$. Пусть $G_i = (V_{N_i}, V_{T_i}, P_i, S_i)$ — грамматика, порождающая множество $f(a_i)$ для каждого i, $1 \leq i \leq n$. Без потери общности предполагаем, что все нетерминальные словари попарно не пересекаются.

Докажем теорему для случая, когда грамматики G и G_i, $1 \leq i \leq n$, являются контекстно-свободными. Читатель может доказать другие случаи аналогично, хотя в каждом необходимы дополнительные детали.

Построим новую грамматику:

$$G' = (V_N', V_T', P', S), \text{ где } V_N' = V_N \cup \bigcup_{i=1}^{n} V_{N_i}, \; V_T' = \bigcup_{i=1}^{n} V_{T_i}.$$

Пусть h — подстановка $h(a_i) = \{S_i\}$ для $1 \leq i \leq n$ и $h(A) = \{A\}$ для любого $A \in V_N$; $P' = \bigcup_{i=1}^{n} P_i \cup \{A \to h(\alpha) \mid A \to \alpha \in P\}$. Ясно, что грамматика G' является контекстно-свободной, возможно, с правилами вида $A \to \varepsilon$. Очевидно, что $f(L(G)) = L(G')$.

Пример 9.2. Пусть $L = \{0^n 1^n \mid n \geq 1\}$. Язык L порождается грамматикой

$$G = (\{S\}, \{0, 1\}, \{S \to 0S1, S \to 01\}, S).$$

Как и в примере 9.1, пусть

$$f(0) = \{a\} \text{ и } f(1) = \{ww^R \mid w \in \{b, c\}^*\};$$

$f(0)$ порождается грамматикой

$$G_1 = (\{S_1\}, \{a\}, \{S \to a\}, S_1),$$

а $f(1)$ — грамматикой

$$G_2 = (\{S_2\},\ \{\mathrm{b},\ \mathrm{c}\},\ \{S_2 \to bS_2b,\ S_2 \to cS_2c,\ S_2 \to \varepsilon\},\ S_2).$$

Язык $f(L)$ порождается грамматикой $G_3 = (\{S, S_1, S_2\},\ \{a, b, c\},\ \{S \to S_1SS_2,\ S \to S_1S_2,\ S_1 \to a,\ \ S_2 \to bS_2b,\ S_2 \to cS_2c,\ S_2 \to \varepsilon\},\ S)$. Первые два правила грамматики G_3 получились из правил $S \to 0S1$ и $S \to 01$ грамматики G_1 в результате подстановки символа S_1 вместо 0 и символа S_2 — вместо 1.

Контекстно-зависимые языки не замкнуты относительно подстановки. Однако мы можем несколько смягчить этот факт.

***Определение* 9.2.** Подстановка f называется *ε-свободной* (ε-free), если $\varepsilon \notin f(a)$ для каждого $a \in \Sigma$.

Теорема 9.8. *Класс контекстно-зависимых языков замкнут относительно* ε*-свободной подстановки*.

Доказательство. Рассмотрим контекстно-зависимую грамматику $G = (V_N,\ \{a_1,\ a_2, \ldots,\ a_n\},\ P,\ S)$ и ε-свободную подстановку f. Пусть для каждого i, $1 \le i \le n$, $G_i = (V_{N_i}, V_{T_i}, P_i, S_i)$ — контекстно-зависимая грамматика, порождающая множество $f(a_i)$. Без потери общности предполагаем, что все нетерминальные словари попарно не пересекаются. Кроме того, предполагаем, что все правила, за возможным исключением $S \to \varepsilon$, имеют вид $\alpha \to \beta$ или $A \to a$, где α, β — непустые строки нетерминалов, A — отдельный нетерминал, а a — отдельный терминальный символ. Мы построим грамматику $G' = (V_N', V_T', P', S_L)$, где

1. $V_N' = V_N \cup \bigcup_{i=1}^{n} V_{N_i} \cup \{A_L \mid A \in V_N\}$.

2. $V_T' = \bigcup_{i=1}^{n} V_{T_i}$.

3. P' содержит

а) $S_L \to \varepsilon$, если $S \to \varepsilon \in P$;

б) $A_L\alpha \to B_L\beta$ и $A\alpha \to B\beta$, если $A\alpha \to B\beta \in P$ (заметим, что индекс L в обозначении A_L помечает самое левое вхождение соответствующего нетерминального символа в выводе в грамматике G до тех пор, пока этот символ не превратится в терминальный символ);

в) $A_L \to S_i$, если $A \to a_i \in P$,

$aA \to aS_i$ для всех $a \in V_T'$, если $A \to a_i \in P$;

г) все правила из множества $\{P_i \mid i = 1, 2, \ldots, n\}$.

Грамматика G' — контекстно-зависимая и $L(G') = f(L(G))$.

Теорема 9.9. *Класс контекстно-зависимых языков не замкнут относительно подстановки*.

Доказательство. Пусть $G_1 = (V_N, V_T, P_1, S)$ — грамматика типа 0, такая, что $L(G_1)$ не является контекстно-зависимым языком. Снова мы предполагаем без потери общности, что все ее правила имеют вид $\alpha \to \beta$ или $A \to a$, где $\alpha \in V_N^+$, $\beta \in V_N^*$, $A \in V_N$, $a \in V_T$.

Пусть c — новый символ. Рассмотрим грамматику $G_2 = (V_N, V_T \cup \{c\}, P_2, S)$, в которой P_2 содержит

1) $\alpha \to \beta$, если $\alpha \to \beta \in P_1$ и $|\alpha| \le |\beta|$;

2) $\alpha \to \beta cc\ldots c$, где $|\alpha| = |\beta cc\ldots c|$, если $\alpha \to \beta \in P_1$ и $|\alpha| > |\beta|$;

3) $cA \to Ac$ для всех $A \in V_N$.

Грамматика G_2 является контекстно-зависимой, поскольку мы принудили правую часть каждого правила иметь, по крайней мере, такую же длину, как левая. Правила $cA \to Ac$ были добавлены для того, чтобы передвигать символы c к правому концу слов так, чтобы выводы в G_2 могли происходить, как в грамматике G_1.

Теперь рассмотрим подстановку $f(a) = \{a\}$ для $a \in V_T$ и $f(c) = \{\varepsilon\}$. Тогда $f(L(G_2)) = L(G_1)$ и, следовательно, подстановка не сохраняет класс csl.

Очень часто интерес представляют подстановки специальных типов.

***Определение* 9.3.** Подстановка f называется *конечной*, если $f(a)$ есть конечное множество для всех a из области определения f. Если $f(a)$ — единственная строка, то f — *гомоморфизм*.

Конечная подстановка и гомоморфизм являются специальными классами подстановок. Из этого мы имеем следующие следствия:

***Следствие* 9.1.** Классы регулярных, контекстно-свободных и языков типа 0 замкнуты относительно конечной подстановки и гомоморфизма.

Доказательство очевидно из теоремы 9.7.

***Следствие* 9.2.** Класс контекстно-зависимых языков замкнут относительно ε-свободной конечной подстановки и ε-свободного гомоморфизма.

Доказательство очевидно из теоремы 9.8.

***Следствие* 9.3.** Класс контекстно-зависимых языков не замкнут относительно конечной подстановки и гомоморфизма.

Доказательство. Подстановка, использованная при доказательстве теоремы 9.9, является гомоморфизмом.

Мы докажем еще один результат, касающийся подстановок, поскольку он необходим для последующей теоремы.

***Определение* 9.4.** Класс языков замкнут относительно *k-ограниченного стирания*, если для любого языка L этого класса и любого гомоморфизма h, обладающего тем свойством, что h никогда не отображает более, чем k последовательных символов любого предложения из языка L в ε, $h(L)$ находится в этом же классе.

Покажем, что класс контекстно-зависимых языков замкнут относительно k-ограниченного стирания. Фактически справедливо более общее утверждение. Пусть $L \subseteq \Sigma^*$ — контекстно-зависимый язык и пусть $f(a)$ для любого $a \in \Sigma$ тоже контекстно-зависим. Тогда язык $f(L)$ контекстно-зависим при условии, что существует $k > 0$, такое, что для $x \in L$ и $y \in f(x)$ выполняется неравенство $|y| \ge k|x|$.

Лемма 9.2. *Класс контекстно-зависимых языков замкнут относительно k-ограниченного стирания.*

Доказательство. Пусть $G_1 = (V_N^{(1)},\ V_T^{(1)},\ P_1,\ S_1)$ — контекстно-зависимая грамматика. Без потери общности предположим, что правила, за возможным исключением $S_1 \to \varepsilon$, имеют вид $\alpha \to \beta$ или $A \to a$, где $\alpha, \beta \in V_N^{(1)+}$, $A \in V_N^{(1)}$, а $a \in V_T^{(1)}$. Пусть h — гомоморфизм со свойством, что h никогда не отображает более, чем k последовательных символов любого предложения $x \in L(G_1)$ в ε. Пусть целое l больше, чем $k+1$, и больше длины самой длинной левой части любого правила. Рассмотрим грамматику

$$G_2 = (V_N^{(2)}, V_T^{(2)}, P_2, S_2),$$

где

$$V_N^{(2)} = \{[\alpha] \mid \alpha \in (V_N^{(1)} \cup V_T^{(1)})^*, |\alpha| < 2l\},$$

$V_T^{(2)}$ содержит такие символы, находящиеся в строках w, что $h(a) = w$ для некоторого $a \in V_T^{(1)}$, $S_2 = [S_1]$, а множество правил P_2 содержит

1) $[S_1] \to \varepsilon$, если $S_1 \to \varepsilon \in P_1$ или если $x \in L(G_1)$ и $h(x) = \varepsilon$ (заметим, что $|x| \le k$, так что мы можем проверить, существует ли какая-нибудь такая цепочка x);

2) $[\alpha] \to [\beta]$ для всех $[\alpha]$ и $[\beta]$ из $V_N^{(2)}$, таких, что $\alpha \underset{G_1}{\Rightarrow} \beta$ и $|\beta| < 2l$;

3) $[\alpha] \to [\beta_1][\beta_2]\ldots[\beta_m]$ для всех $[\alpha]$, $[\beta_1]$, $[\beta_2]$,…, $[\beta_m]$ из $V_N^{(2)}$, таких, что $\alpha \underset{G_1}{\Rightarrow} \beta_1\beta_2\ldots\beta_m$, $|\beta_i| = l$, $1 \le i < m$, $l \le |\beta_m| < 2l$;

4) $[\alpha_1][\alpha_2] \to [\beta_1][\beta_2]\ldots[\beta_m]$ для всех $[\alpha_1]$, $[\alpha_2]$, $[\beta_1]$, $[\beta_2]$,…, $[\beta_m]$ из $V_N^{(2)}$, таких, что $\alpha_1\alpha_2 \underset{G_1}{\Rightarrow} \beta_1\beta_2\ldots\beta_m$, $l \le |\alpha_1| < 2l$, $l \le |\alpha_2| < 2l$, $|\beta_i| = l$, $1 \le i < m$, $l \le |\beta_m| < 2l$;

5) $[x] \to h(x)$ для всех $[x] \in V_N^{(2)}$, $x \in V_T^{(1)*}$, $h(x) \ne \varepsilon$.

Грамматика G_2 является контекстно-зависимой и $L(G_2) = h(L(G_1))$. Отметим, что G_2 получается путем кодирования блоков по меньшей мере из $k+1$ символа грамматики G_1 в один символ. Поскольку не более k последовательных терминальных символов грамматики G_1 отображаются в ε, то в грамматике G_2 никогда не требуется иметь правило, в котором нетерминал, не равный начальному, порождал бы ε.

***Определение* 9.5.** *Обобщенная последовательная машина* (gsm[10]) есть конечный автомат, который может выводить конечное число символов для каждого входного символа. Формально обобщенная последовательная машина есть система $S = (Q, \Sigma, \Delta, \delta, q_0, F)$, где Q — *состояния*; Σ — *входной алфавит*; Δ — *выходной алфавит*; δ — *отображение типа* $Q \times \Sigma \to 2^{Q \times \Delta^*}$; $q_0 \in Q$ — *начальное*

[10] Gsm — generalized sequential machine. В качестве синонима в настоящее время используется более современный термин *конечный преобразователь* (finite transducer — ft).

состояние; $F \subseteq Q$ — *множество конечных состояний*. Запись $(p, w) \in \delta(q, a)$ означает, что S в состоянии q, имея на входе символ a, может в качестве одного из возможных вариантов движения перейти в состояние p и вывести строку w.

Мы расширим область определения δ до $Q \times \Sigma^*$ следующим образом:

$\delta(q, \varepsilon) = \{(q, \varepsilon)\}$,

$\delta(q, xa) = \{(p, w) \mid w = w_1 w_2,\ (p', w_1) \in \delta(q, x)$ и $(p, w_2) \in \delta(p', a)\}$, если $x \in \Sigma^*$ и $a \in \Sigma$.

***Определение* 9.6.** Пусть S — обобщенная последовательная машина и $S(x) = \{y \mid (p, y) \in \delta(q_0, x)$ для некоторого $p \in F\}$. Если L есть язык над Σ, то $S(L) = \{y \mid y \in S(x)$ для некоторого $x \in L\}$ называется *gsm-отображением*, а $S^{-1}(L) = \{y \mid x \in S(y)$ для некоторого $x \in L\}$ — *обратным gsm-отображением*.

Не обязательно истинно, что $S^{-1}(S(L)) = S(S^{-1}(L)) = L$, и потому отображение S^{-1} не является подлинно обратным

Пример 9.3. Пусть $S = (\{q_0, q_1\}, \{0, 1\}, \{a, b\}, \delta, q_0, \{q_1\})$ — обобщенная последовательная машина, где отображение δ определено следующим образом:

1) $\delta(q_0, 0) = \{(q_0, aa), (q_1, b)\}$,
2) $\delta(q_0, 1) = \{(q_0, a)\}$,
3) $\delta(q_1, 0) = \varnothing$,
4) $\delta(q_1, 1) = \{(q_1, \varepsilon)\}$.

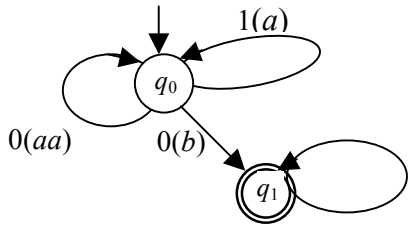

Рис. 9.2.

Интуитивно, пока в gsm S вводятся нули (рис. 9.2), gsm S имеет выбор: выводить два символа a либо один символ b. Если gsm S выводит b, она переходит в состояние q_1. Если gsm S находится в состоянии q_0 и в нее вводится символ 1, то она может выводить только символ a. В состоянии q_1 gsm S ничего не может поделать с 0 на входе, но может оставаться в состоянии q_1 без какого-либо вывода, если на входе 1.

Пусть $L = \{0^n 1^n \mid n \geq 1\}$. Тогда $S(L) = \{a^{2n} b \mid n \geq 0\}$.

Если обозначить $S(L)$ при помощи L_1, то $S^{-1}(L_1) = \{w01^i \mid i \geq 0$ и w имеет[11] четное число $1\}$. Заметим, что $S^{-1}(S(L)) \neq L$.

Характерная особенность gsm-отображения и обратного gsm-отображения состоит в том, что они сохраняют разные классы языков.

[11] При этом w может содержать любое число нулей.

Лемма 9.3. *Каждый класс языков, замкнутый относительно конечной подстановки и пересечения с регулярным множеством, замкнут относительно* gsm-*отображений.*

Доказательство. Пусть C — класс языков, замкнутый относительно конечной подстановки (следовательно, также и гомоморфизма) и пересечения с регулярным множеством. Пусть $S = (Q, \Sigma, \Delta, \delta, q_0, F)$ — обобщенная последовательная машина. Определим конечную подстановку

$f(a) = \{[q, a, x, p] \mid q, p \in Q, a \in \Sigma, x \in \Delta^*,$ и $(p, x) \in \delta(q, a)\}$.

Пусть R — регулярное множество, содержащее все строки вида

$$[q_0, a_1, x_1, q_1]\,[q_1, a_2, x_2, q_2] \ldots [q_{n-1}, a_n, x_n, q_n],$$

такие, что для $1 \le i \le n$, $a_i \in \Sigma$, $x_i \in \Delta^*$, $q_i \in Q$, $(q_i, x_i) \in \delta(q_{i-1}, a_i)$. Также q_0 — начальное состояние и $q_n \in F$. Пусть $h([q, a, x, p]) = x$ для всех $[q, a, x, p]$.

Теперь для $L \in C$ имеем $S(L) = h(f(L) \cap R)$. Поскольку класс языков C замкнут относительно конечной подстановки и пересечения с регулярным множеством, то язык $S(L)$ тоже находится в C. Заметим, что требуется замкнутость относительно конечной подстановки, а не ε-свободной конечной подстановки, поскольку в $[q, a, x, p]$ цепочка x может быть равна ε, и в этом случае $h([q, a, x, p]) = \varepsilon$.

Теорема 9.10. *Классы регулярных, контекстно-свободных и языков типа 0 замкнуты относительно gsm-отображений.*

Доказательство. Теорема является прямым следствием леммы 9.3 и теорем 9.4, 9.6 и 9.7.

Отметим, что gsm-отображения не сохраняют контекстно-зависимых языков, поскольку каждый гомоморфизм является gsm-отображением.

***Определение* 9.7.** Говорят, что gsm-отображение ε-*свободно*, если $(p,\varepsilon) \notin \delta(q, a)$ для любых $q, p \in Q$ и $a \in \Sigma$.

Хотя контекстно-зависимые языки не замкнуты относительно произвольных gsm-отображений, они замкнуты относительно ε-свободных gsm-отображений.

Теорема 9.11. *Класс контекстно-зависимых языков замкнут относительно ε-свободных gsm-отображений.*

Доказательство. В лемме 9.3 конечная подстановка может быть заменена на ε-свободную конечную подстановку при условии, что gsm-отображение ε-свободно. Таким образом, поскольку класс контекстно-зависимых языков замкнут относительно ε-свободной конечной подстановки и пересечения с регулярным множеством, то этот класс замкнут относительно ε-свободных gsm-отображений.

Рассмотрим теперь обратные gsm-отображения. Как увидим, регулярные, контекстно-свободные, контекстно-зависимые и языки типа 0 все замкнуты относительно обратных gsm-отображений.

Лемма 9.4. *Пусть C — класс языков, замкнутый относительно ε-свободной подстановки, k-ограниченного стирания и объединения и пересечения с регулярными множествами. Тогда класс C замкнут относительно обратных gsm-отображений.*

Доказательство. Пусть $L \subseteq \Delta^*$ есть язык в классе C, а $S = (Q, \Sigma, \Delta, \delta, q_0, F)$ — обобщенная последовательная машина. Мы предполагаем без потери общности, что $\Sigma \cap \Delta = \varnothing$. Определим подстановку f следующим образом: $f(b) = \Sigma^* b$ для каждого $b \in \Delta$. (Отметим, что замкнутость относительно объединения и пересечения с регулярными множествами гарантирует принадлежность всех регулярных множеств классу C и, следовательно, $\Sigma^* b \in C$.)

Пусть $L_1 = f(L) \cup \Sigma^*$, если $\varepsilon \in L$, и $L_1 = f(L)$ в противном случае. Тогда L есть множество всех строк вида $y_1 b_1 y_2 b_2 \ldots y_r b_r$, $r \geq 1$, где $b_i \in \Delta$, $y_i \in \Sigma^*$, $1 \leq i \leq r$, $b_1 b_2 \ldots b_r \in L$, объединенное с Σ^*, если $\varepsilon \in L$. Применим теперь лемму 9.3 к классам регулярных, контекстно-свободных и языков типа 0.

Пусть R — регулярное множество, состоящее из всех слов вида $a_1 x_1 a_2 x_2 \ldots a_m x_m$, $m \geq 0$, таких, что

1) $a_i \in \Sigma$;

2) $x_i \in \Delta^*$, $1 \leq i \leq m$.

Существуют состояния $q_0, q_1, \ldots, q_m$, такие, что $q_m \in F$ и $(q_i, x_i) \in \delta(q_{i-1}, a_i)$ для $1 \leq i \leq m$.

Заметим, что цепочка x_i может быть равна ε. Нетрудно показать путем построения конечного автомата, принимающего R, что R — регулярное множество.

Теперь $L_1 \cap R$ есть множество всех слов вида $a_1 x_1 a_2 x_2 \ldots a_m x_m$, $m \geq 0$, где $a_i \in \Sigma$, $x_i \in \Delta^*$, $1 \leq i \leq m$, $x_1 x_2 \ldots x_m \in L$, $S(a_1 a_2 \ldots a_m)$ содержит цепочку $x_1 x_2 \ldots x_m$, и ни одна цепочка x_i не длиннее, чем k, причем k — длина самой длинной цепочки x, такой, что $(p, x) \in \delta(q, a)$ для некоторых состояний $p, q \in Q$ и $a \in \Sigma$.

Наконец, пусть h — гомоморфизм, который отображает символ a в a для каждого $a \in \Sigma$ и символ b — в ε для каждого $b \in \Delta$. Тогда $S^{-1}(L) = h(L_1 \cap R)$ находится в классе C, поскольку h никогда не отображает больше k последовательных символов в ε.

Теорема 9.12. Классы регулярных, контекстно-свободных, контекстно-зависимых и языков типа 0 замкнуты относительно обратных gsm-отображений.

Доказательство следует непосредственно из леммы 9.4 и того факта, что названные классы замкнуты относительно ε-свободной подстановки, k–ограниченного стирания, а также пересечения и объединения с регулярным множеством.

Теперь рассмотрим операцию деления.

***Определение* 9.8.** Пусть L_1 и L_2 — любые два языка. Определим *частное от деления L_1 на L_2* как множество $\{x \mid$ для некоторой цепочки $y \in L_2$, такой, чтобы $xy \in L_1\}$.

Пример 9.4. Пусть $L_1 = \{a^n b^n \mid n \geq 1\}$ и $L_2 = b^*$. Тогда

$$L_1 / L_2 = \{a^i b^j \mid i \geq j,\ i \geq 1\},\ \text{а}\ L_2 / L_1 = \varnothing.$$

Лемма 9.5. *Каждый класс языков, замкнутый относительно конечной подстановки и пересечения с регулярным множеством, замкнут относительно деления на регулярное множество*.

Доказательство. Пусть C — класс языков, замкнутый относительно названных операций. Пусть $L \in \Sigma_1^*$ — язык из класса C и $R \subseteq \Sigma_1^*$ — регулярное множество. Пусть $\Sigma_2 = \{a' \mid a \in \Sigma_1\}$ и f — конечная подстановка: $f(a) = \{a, a'\}$. Рассмотрим $L_2 = \Sigma_2^* R \cap f(L)$. Пусть h — гомоморфизм, определяемый следующим образом: $h(a) = \varepsilon$ и $h(a') = a$ для всех $a \in \Sigma_1$. Теперь $L / R = h(L_2)$. Поскольку класс C замкнут относительно конечной подстановки и пересечения с регулярным множеством, то L / R находится в классе C.

Теорема 9.13. *Классы регулярных, контекстно-свободных и языков типа 0 замкнуты относительно деления на регулярное множество*.

Доказательство следует непосредственно из леммы 9.5.

На вопрос: замкнут ли класс контекстно-зависимых языков относительно деления на регулярное множество, ответим — нет.

Теорема 9.14. *Если L_1 есть любой язык типа* 0, *то существует контекстно-зависимый язык L_2 и регулярное множество R, такие, что $L_1 = L_2 / R$.*

Доказательство почти идентично доказательству теоремы 9.9. Пусть $G_1 = (V_N, V_T, P_1, S_1)$ — грамматика типа 0, такая, что $L(G_1) = L_1$ и пусть $G_2 = (V_N \cup \{S_1, D\}, V_T \cup \{c, d\}, P_2, S_2)$, где P_2 определяется следующим образом:

1) если $\alpha \to \beta \in P_1$ и $|\alpha| \leq |\beta|$, то $\alpha \to \beta \in P_2$;

2) если $\alpha \to \beta \in P_1$ и $|\alpha| - |\beta| = i$, $i > 0$, то $\alpha \to \beta D^i \in P_2$;

3) для всех $A \in V_N$ и $a \in V_T$ существуют правила $DA \to AD$ и $Da \to aD \in P_2$;

4) существуют правила $Dc \to cc$ и $Dc \to dc \in P_2$;

5) существует правило $S_2 \to S_1 Dc \in P_2$.

Обратим внимание читателя на сходство $L(G_2)$ с языком, определенным в теореме 9.9. Но здесь мы можем превращать все нетерминалы D в терминальные символы, если только они сначала мигрируют к правому концу сентенциальной формы. Причем как только нетерминал D превращается в терминал d, ни один символ D больше не может быть превращен ни в d, ни в c. Теорема следует из наблюдения, что $L(G_1) = L(G_2) / dc^*$.

В заключение главы приведем сводку описанных в ней свойств замкнутости для регулярных, контекстно-свободных, контекстно-зависимых и языков типа 0, — см. табл. 9.1.

Табл. 9.1

Замкнутость относительно операций	Класс языка			
	рег.	конт.-свобод.	конт.-завис.	типа 0
Объединение	+	+	+	+
Конкатенация	+	+	+	+
Замыкание	+	+	+	+
Обращение	+	+	+	+
Пересечение	+	–	+	+
Дополнение	+	–	?	–
Пересечение с регулярным множеством	+	+	+	+
Подстановка	+	+	+	+
ε-Свободная подстановка	+	+	+	+
gsm-Отображение	+	+	+	+
ε-Свободное gsm-отображение	+	+	+	+
Обратные gsm-отображения	+	+	+	+
k-Ограниченное стирание	+	+	+	+
Деление на регулярное множество	+	+	+	+

В табл. 9.1 символом + отмечен тот факт, что соответствующий класс языков обладает свойством замкнутости относительно соответствующей операции; символ – (минус) обозначает отсутствие соответствующего свойства замкнутости для соответствующего класса языков; символ ? (вопросительный знак) означает, что пока не выяснено, замкнут ли класс контекстно-зависимых языков относительно дополнения или не замкнут.

Глава 1

ТРАНСЛЯЦИИ, ИХ ПРЕДСТАВЛЕНИЕ И РЕАЛИЗАЦИЯ

§ 1.1. Трансляции и трансляторы

***Определение* 1.1.** *Трансляцией* из языка $L_1 \subseteq \Sigma^*$ в язык $L_2 \subseteq \Delta^*$ называется отношение $\tau \subseteq L_1 \times L_2$. Здесь Σ — *входной алфавит*, L_1 — *входной язык*, Δ — *выходной алфавит*, L_2 — *выходной язык*.

Другими словами, трансляция есть некоторое множество пар предложений (x, y), где $x \in L_1$ — *входное*, а $y \in L_2$ — *выходное предложение*. Хотя в общем случае в трансляции τ одному входному предложению x может соответствовать несколько выходных предложений y, по отношению к языкам программирования трансляция всегда является функцией, т.е. для каждого входа существует не более одного выхода.

Существует бесконечно много примеров трансляций, но самым элементарным, вероятно, является тот, который может быть задан *гомоморфизмом*, т.е. отображением h из Σ в Δ^*.

Пример 1.1. Предположим, что мы хотим закодировать некоторый текст с помощью азбуки Морзе. Как известно, в коде Морзе каждая буква представляется как некоторая последовательность из точек и тире. Эти последовательности, называемые *посылками*, имеют разную длину. Для отделения одной посылки от другой используется *пауза*. Очевидно, что трансляцию предложений, например на русском языке, в код Морзе можно реализовать с помощью гомоморфизма, задаваемого следующим образом:

Буква	а	б	в	…	я
Посылка	· —	— ···	· — —	…	· — · —

Для простоты предполагаем, что паузы представлены пробелами, завершающими каждую посылку. Тогда, скажем, слово “абба” с помощью замены букв на посылки даст результат:

· — — ··· — ··· · —

Для любой входной цепочки $x = a_1 a_2 \ldots a_n$, $a_i \in \Sigma$, $i = 1, 2, \ldots, n$, гомоморфизм h позволяет найти соответствующую выходную цепочку y с помощью посимвольной подстановки: $y = h(a_1)h(a_2)\ldots h(a_n)$.

Область определения гомоморфизма можно расширить до Σ^* следующим образом: $h(ax) = h(a)h(x)$, $h(\varepsilon) = \varepsilon$. Здесь $a \in \Sigma$, $x \in \Sigma^*$.

Гомоморфизм h определяет трансляцию $\tau(h) = \{(x,\ h(x)) \mid x \in \Sigma^*\}$.

Устройство, которое по заданной цепочке $x \in \Sigma^*$ находит соответствующую цепочку $y = h(x)$, такую, что $(x,y) \in \tau(h)$, тривиально: оно должно посимвольно просмотреть входную цепочку x и заменить каждый ее символ a на $h(a)$. Это устройство является примером простейшего транслятора, реализующего трансляцию $\tau(h)$.

Реалистичным примером транслятора, основанного на гомоморфизме, является простейший *ассемблер*.

Транслятором для данной трансляции τ называется такое устройство, которое по данной входной строке x вычисляет выходную цепочку y, такую, что $(x,\ y) \in \tau$. Желательными свойствами транслятора являются

1) *эффективность* (время, затрачиваемое на перевод входной строки, должно быть линейно пропорционально ее длине);

2) малый размер;

3) *правильность* (желательно иметь небольшой тест, такой, чтобы если транслятор прошел его, то это гарантировало бы правильную работу транслятора на *всех* входных цепочках).

§ 1.2. Схемы синтаксически управляемой трансляции

Трансляторы являются средством реализации трансляций, хотя их можно рассматривать также и как способ их задания.

Способом спецификации трансляций, более сложных, чем те, которые описываются при помощи гомоморфизма, является аппарат *схем синтаксически управляемых трансляций* (sdts — syntax-directed translation schema).

***Определение* 1.2.** *Схемой синтаксически управляемой трансляции* называется формальная система $T = (N, \Sigma, \Delta, R, S)$, где N — *алфавит нетерминалов*; Σ — конечный *входной алфавит*; Δ — конечный *выходной алфавит*, причем $N \cap \Sigma = \varnothing$ и $N \cap \Delta = \varnothing$; R — конечное *множество правил схемы* вида $A \to \alpha, \beta$, где $A \in N$, $\alpha \in (N \cup \Sigma)^*$, $\beta \in (N \cup \Delta)^*$, причем каждое вхождение нетерминала в цепочку α взаимно-однозначно связано с некоторым вхождением одноименного нетерминала в β, и эта связь является неотъемлемой частью правила; $S \in N$ — начальный нетерминал.

Пусть $A \to \alpha, \beta$ — правило схемы. Цепочка α называется *синтаксической*, а β — *семантической*.

Для указания связей между вхождениями нетерминалов можно использовать индексы. Связанные вхождения нетерминалов помечаются одинаковыми индексами. Например, $A \to aB^{(1)}bCB^{(2)}$, $B^{(2)}B^{(1)}dC$.

***Определение* 1.3.** Введем понятие *трансляционной формы* следующим образом:

1) (S, S) — *начальная трансляционная форма*, причем эти два вхождения начального нетерминала связаны друг с другом по определению;

2) если $(\alpha A\beta, \alpha' A\beta')$ — трансляционная форма, в которой два явно выделенных вхождения нетерминала A связаны, и если $A \to \gamma, \gamma'$ — правило из R, то $(\alpha\gamma\beta, \alpha'\gamma'\beta')$ — трансляционная форма; причем связь между нетерминалами в γ и γ' — такая же, как в правиле; нетерминалы в цепочках α и β связываются с нетерминалами в цепочках α' и β' в новой трансляционной форме точно так же, как в предыдущей; *связь между нетерминалами* является существенной чертой трансляционной формы;

3) трансляционными формами являются такие и только такие пары цепочек, которые могут быть получены с помощью данных двух способов.

Это определение фактически вводит отношение непосредственной выводимости одной трансляционной формы из другой. В таком случае принято писать: $(\alpha A\beta, \alpha' A\beta') \underset{T}{\Rightarrow} (\alpha\gamma\beta, \alpha'\gamma'\beta')$. Для степени, транзитивного и рефлексивно-транзитивного замыкания этого отношения используются соответственно следующие обозначения: $\overset{k}{\underset{T}{\Rightarrow}}$, $\overset{+}{\underset{T}{\Rightarrow}}$ и $\overset{*}{\underset{T}{\Rightarrow}}$. Когда ясно, в какой схеме производится вывод, имя схемы может быть опущено.

***Определение* 1.4.** *Трансляция, заданная при помощи схемы синтаксически управляемой трансляции* T, есть множество $\tau(T) = \{(x, y) \mid (S, S) \overset{*}{\underset{T}{\Rightarrow}} (x, y), x \in \Sigma^*, y \in \Delta^*\}$ и называется *синтаксически управляемой трансляцией* (sdt).

***Определение* 1.5.** Грамматика $G_i = (N, \Sigma, P_i, S)$, где $P_i = \{A \to \alpha \mid \exists A \to \alpha, \beta \in R\}$, называется *входной грамматикой схемы*. Грамматика $G_o = (N, \Delta, P_o, S)$, где $P_o = \{A \to \beta \mid \exists A \to \alpha, \beta \in R\}$, называется *выходной грамматикой схемы*.

Очевидно, что G_i и G_o — контекстно-свободные грамматики.

Пример 1.2. Пусть sdts $T = (\{E, T, F\}, \{a, +, *, (,)\}, \{a, +, *\}, R, E)$, где

$R = \{(1)\ E \to E+T, ET+;\ (2)\ E \to T, T;\ (3)\ T \to T*F, TF*;$

$(4)\ T \to F, F;\ (5)\ F \to (E), E;\ (6)\ F \to a, a\}.$

Связь между нетерминалами в этих правилах очевидна, так как в синтаксической и семантической цепочках каждого правила нет двух или более вхождений одноименных нетерминалов. Рассмотрим какой-нибудь вывод в T, например:

$$(E, E) \Rightarrow (E + T, ET +) \Rightarrow (T^{(1)} + T^{(2)}, T^{(1)}T^{(2)} +) \Rightarrow (F^{(1)} + T^{(2)}, F^{(1)}T^{(2)} +) \Rightarrow$$
$$\Rightarrow (a + T, aT +) \Rightarrow (a + F^{(1)} * F^{(2)}, aF^{(1)}F^{(2)} * +) \Rightarrow$$
$$\Rightarrow (a+a*F, aaF*+) \Rightarrow (a+a*a, aaa*+).$$

Нетрудно догадаться, что $\tau(T) = \{(x, y) \mid x$ — инфиксная запись арифметического выражения, y — эквивалентная постфиксная запись$\}$.

***Определение* 1.6.** Схема синтаксически управляемой трансляции называется *простой*, если в каждом ее правиле $A \to \alpha, \beta$ связанные нетерминалы в цепочках α и β встречаются в одинаковом порядке.

Трансляция, определяемая простой схемой, называется *простой синтаксически управляемой трансляцией*.

Многие, но не все, полезные трансляции могут быть описаны как простые. В примере 1.2 схема T, как и определяемая ею трансляция $\tau(T)$, является простой.

Простые синтаксически управляемые трансляции интересны тем, что каждая из них может быть реализована транслятором в классе недетерминированных магазинных преобразователей (рис. 1.1).

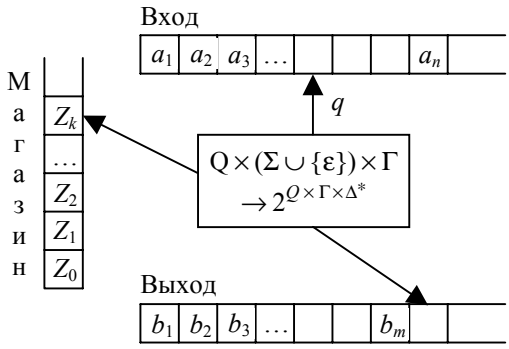

Рис. 1.1.

Другими словами, *магазинные преобразователи характеризуют класс простых синтаксически управляемых трансляций* таким же образом, как магазинные автоматы характеризуют класс контекстно-свободных языков. К рассмотрению таких трансляций мы сейчас и перейдем.

§ 1.3. Магазинные преобразователи и синтаксически управляемые трансляции

В гл. 9 ч. I были описаны обобщенные последовательные машины, называемые также *конечными преобразователями*. Здесь мы рассмотрим магазинные преобразователи, отличающиеся от конечных, лентой памяти, используемой в режиме магазина, как в магазинном автомате.

***Определение* 1.7.** *Магазинный преобразователь* (pdt — pushdown transducer) есть формальная система $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F)$, где Q — конечное *множество состояний*, Σ — конечный *входной алфавит*, Γ — конечный *алфавит магазинных символов*, Δ — конечный *выходной алфавит*, $q_0 \in Q$ — *начальное состояние*, $Z_0 \in \Gamma$ — *начальный символ магазина*, $F \subseteq Q$ — множество *конечных состояний*, δ — отображение типа: $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma \times \Delta^*}$.

Запись $\delta(q, a, Z) = \{(p_i, \gamma_i, y_i)\}_{i=1}^{n}$ означает, что pdt P, находясь в состоянии $q \in Q$, сканируя символ $a \in \Sigma \cup \{\varepsilon\}$ на входной ленте и имея $Z \in \Gamma$ на вершине магазина, переходит в одно из состояний $p_i \in Q$, заменяя в магазине символ Z на

$\gamma_i \in \Gamma^*$ и записывая $y_i \in \Delta^*$ на выходную ленту. При этом входная головка сдвигается на одну ячейку вправо, если $a \neq \varepsilon$ (иначе головка остается на месте), головка магазина сканирует последнюю запись в магазине, а головка выходной ленты размещается справа от последней ее записи.

В частности: если $a = \varepsilon$, то выбор действия не зависит от текущего входного символа и, как уже отмечалось, входная головка неподвижна. Если $\gamma_i = \varepsilon$, то верхний символ магазина стирается. Если $y_i = \varepsilon$, то на выходную ленту ничего не пишется, и ее головка остается на прежнем месте.

В начальный момент $q = q_0$, в магазине находится единственный символ Z_0, входная головка сканирует первый входной символ, а выходная лента пуста, причем ее головка находится на первой ячейке.

Работу магазинного преобразователя опишем в терминах конфигураций.

***Определение* 1.8.** *Конфигурацией магазинного преобразователя P* назовем четверку (q, x, α, y), где $q \in Q$ — *текущее состояние*, $x \in \Sigma^*$ — часть входной ленты от текущего символа до конца, называемая *непросмотренной частью входной цепочки*, $\alpha \in \Gamma^*$ — *содержимое магазина* (будем считать, что первый символ цепочки α есть *верхний символ магазина*), $y \in \Delta^*$ — *содержимое выходной ленты*. Таким образом, начальная конфигурация есть $(q_0, x, Z_0, \varepsilon)$, где x обозначает всю входную цепочку.

Пусть $(q, ax, Z\alpha, y)$ — текущая конфигурация и $(p, \gamma, z) \in \delta(q, a, Z)$. Тогда один такт работы pdt P записывается как отношение между двумя последовательными конфигурациями: $(q, ax, Z\alpha, y) \vdash (p, x, \gamma\alpha, yz)$. Здесь $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $x \in \Sigma^*$, $Z \in \Gamma$, $\alpha, \gamma \in \Gamma^*$, $y, z \in \Delta^*$.

Как обычно, определяются степень ($\vdash^k$), транзитивное замыкание ($\vdash^+$) и рефлексивно-транзитивное замыкание ($\vdash^*$) этого отношения.

***Определение* 1.9.** Говорят, что $y \in \Delta^*$ есть *выход* для $x \in \Sigma^*$ *при конечном состоянии*, если $(q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \alpha, y)$ для некоторых $q \in F$ и $\alpha \in \Gamma^*$.

Трансляция, определяемая магазинным преобразователем P *при конечном состоянии*, есть $\tau(P) = \{(x, y) \mid (q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \alpha, y)$ для некоторых $q \in F$ и $\alpha \in \Gamma^*\}$.

Говорят, что $y \in \Delta^*$ есть *выход* для $x \in \Sigma^*$ *при пустом магазине*, если $(q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \varepsilon, y)$ для некоторого $q \in Q$.

Трансляция, определяемая магазинным преобразователем P *при пустом магазине*, есть $\tau_e(P) = \{(x, y) \mid (q_0, x, Z_0, \varepsilon) \vdash^* (q, \varepsilon, \varepsilon, y)$ для некоторого $q \in Q\}$.

Пример 10.3. Пусть $P = (\{q\}, \{a, +, *\}, \{E, +, *\}, \{a, +, *\}, \delta, q, E, \{q\})$, где

1) $\delta(q, a, E) = \{(q, \varepsilon, a)\}$,

2) $\delta(q, +, E) = \{(q, EE+, \varepsilon)\}$,

3) $\delta(q, *, E) = \{(q, EE*, \varepsilon)\}$,

4) $\delta(q, \varepsilon, +) = \{(q, \varepsilon, +)\}$,

5) $\delta(q, \varepsilon, *) = \{(q, \varepsilon, *)\}$.

Рассмотрим действия pdt P на входе $+*aaa$:

$$(q, +*aaa, E, \varepsilon) \vdash (q, *aaa, EE+, \varepsilon) \vdash (q, aaa, EE*E+, \varepsilon) \vdash (q, aa, E*E+, a) \vdash$$
$$\vdash (q, a, *E+, aa) \vdash (q, a, E+, aa*) \vdash (q, \varepsilon, +, aa*a) \vdash (q, \varepsilon, \varepsilon, aa*a+).$$

Данный магазинный преобразователь является примером детерминированного магазинного преобразователя. Очевидно, что он преобразует префиксные арифметические выражения в постфиксные.

***Определение* 1.10.** Магазинный преобразователь $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F)$ называется *детерминированным* (dpdt) , если

1) $\#\delta(q, a, Z) \le 1$ для всех $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$ и $Z \in \Gamma$;

2) если $\delta(q, \varepsilon, Z) \ne \varnothing$ для данных $q \in Q$ и $Z \in \Gamma$, то $\delta(q, a, Z) = \varnothing$ для всех $a \in \Sigma$.

На практике предпочитают использовать dpdt, поскольку в реализации они оказываются более эффективными по сравнению с недетерминированными pdt.

Лемма 1.1. *Пусть $T = (N, \Sigma, \Delta, R, S)$ — простая схема синтаксически управляемой трансляции. Существует недетерминированный магазинный преобразователь P, такой, что $\tau_e(P) = \tau(T)$.*

Доказательство. Построим pdt P, о котором идет речь, и покажем, что он реализует трансляцию $\tau(T)$.

Положим $P = (\{q\}, \Sigma, N \cup \Sigma \cup \Delta', \Delta, \delta, q, S, \varnothing)$. Чтобы отличать в магазине P входные символы от выходных, последние переименовываются с помощью гомоморфизма h, определяемого для каждого выходного символа $b \in \Delta$ при помощи равенства $h(b) = b'$ таким образом, чтобы множество символов $\Delta' = \{b' \mid b \in \Delta\}$ не пересекалось со словарем Σ, т.е. $\Sigma \cap \Delta' = \varnothing$.

Отображение δ определяется так:

1. $(q, x_0 y_0' B_1 x_1 y_1' \dots B_m x_m y_m', \varepsilon)$ включается в $\delta(q, \varepsilon, A)$, если $A \to x_0 B_1 x_1 \dots B_m x_m$, $y_0 B_1 y_1 \dots B_m y_m \in R$, $y_i' = h(y_i)$, $i = 1, 2, \dots, m$, где $m \ge 0$. Здесь $h(by) = b'h(y)$ для каждого $b \in \Delta$ и $y \in \Delta^*$, $h(\varepsilon) = \varepsilon$.

2. $\delta(q, a, a) = \{(q, \varepsilon, \varepsilon)\}$ для всех $a \in \Sigma$.

3. $\delta(q, \varepsilon, b') = \{(q, \varepsilon, b)\}$ для всех $b \in \Delta$.

I. Докажем сначала, что если $(S, S) \overset{*}{\underset{T}{\Rightarrow}} (x, y)$, то $(q, x, S, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$.

Для этого индукцией по длине вывода l докажем более общее утверждение: если для любого $A \in N$ существует вывод $(A, A) \overset{*}{\underset{T}{\Rightarrow}} (x, y)$, то $(q, x, A, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$.

База. Пусть $l = 1$. Имеем $(A, A) \underset{T}{\Rightarrow} (x, y)$. На этом единственном шаге вывода применяется правило $A \to x, y \in R$. Согласно п.1 определения δ имеем $(q, xy', \varepsilon) \in \delta(q, \varepsilon, A)$. Поэтому $(q, x, A, \varepsilon) \underset{P}{\vdash} (q, x, xy', \varepsilon)$.

Далее согласно п.2 $(q, x, xy', \varepsilon) \overset{|x|}{\underset{P}{\vdash}} (q, \varepsilon, y', \varepsilon)$. Наконец, согласно п.3 имеем $(q, \varepsilon, y', \varepsilon) \overset{|y|}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$. Итак, существует переход $(q, x, A, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$.

Индукционная гипотеза. Предположим, что вспомогательное утверждение выполняется для всех выводов длиной $l \le n$ ($n \ge 1$).

Индукционный переход. Докажем утверждение для $l = n + 1$.

Пусть $(A, A) \overset{}{\underset{T}{\Rightarrow}} (x_0 B_1 x_1 \dots B_m x_m, y_0 B_1 y_1 \dots B_m y_m) \overset{n}{\underset{T}{\Rightarrow}} (x, y)$ — вывод длиной $n + 1$.

Очевидно, что $x = x_0 t_1 x_1 t_2 x_2 \dots t_m x_m$, $y = y_0 z_1 y_1 z_2 y_2 \dots z_m y_m$, (1.1)

причем

$$(B_i, B_i) \overset{l_i}{\underset{T}{\Rightarrow}} (t_i, z_i),\ l_i \le n,\ i = 1, 2, \dots, m. \qquad (1.2)$$

На первом шаге данного вывода было применено правило

$$A \to x_0 B_1 x_1 B_2 x_2 \dots B_m x_m, y_0 B_1 y_1 B_2 y_2 \dots B_m y_m \in R$$

и потому согласно п.1 построения имеем

$$(q, x_0 y_0' B_1 x_1 y_1' B_2 x_2 y_2' \dots B_m x_m y_m', \varepsilon) \in \delta(q, \varepsilon, A). \quad (1.3)$$

Кроме того, согласно индукционной гипотезе из существования частичных выводов (1.2), следует возможность перехода

$$(q, t_i, B_i, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, z_i),\ i = 1, 2, \dots, m. \quad (1.4)$$

Рассмотрим движения pdt P. Учитывая условия (1.1) и (1.3), можем написать:

$$(q, x, A, \varepsilon) = (q, x_0 t_1 x_1 t_2 x_2 \dots t_m x_m, A, \varepsilon) \underset{P}{\vdash}$$

$$\underset{P}{\vdash} (q, x_0 t_1 x_1 t_2 x_2 \dots t_m x_m, x_0 y_0' B_1 x_1 y_1' B_2 x_2 y_2' \dots B_m x_m y_m', \varepsilon).$$

Согласно п.2 построений имеем переход

$$(q, x_0 t_1 x_1 t_2 x_2 \dots t_m x_m, x_0 y_0' B_1 x_1 y_1' B_2 x_2 y_2' \dots B_m x_m y_m', \varepsilon) \overset{*}{\underset{P}{\vdash}}$$

$$\overset{*}{\underset{P}{\vdash}} (q, t_1 x_1 t_2 x_2 \dots t_m x_m, y_0' B_1 x_1 y_1' B_2 x_2 y_2' \dots B_m x_m y_m', \varepsilon);$$

согласно п.3 построений имеем переход

$$(q, t_1 x_1 t_2 x_2 \dots t_m x_m, y_0' B_1 x_1 y_1' B_2 x_2 y_2' \dots B_m x_m y_m', \varepsilon) \overset{*}{\underset{P}{\vdash}}$$

$$\overset{*}{\underset{P}{\vdash}} (q, t_1 x_1 \dots t_m x_m, B_1 x_1 y_1' B_2 x_2 y_2' \dots B_m x_m y_m', y_0).$$

Учитывая существование перехода (1.4) для $i = 1$, получаем:

$$(q, t_1 x_1 t_2 x_2 \dots t_m x_m, B_1 x_1 y_1' B_2 x_2 y_2' \dots B_m x_m y_m', y_0) \overset{*}{\underset{P}{\vdash}}$$

$$\overset{*}{\underset{P}{\vdash}} (q, x_1 t_2 x_2 \dots t_m x_m, x_1 y_1' B_2 x_2 y_2' \dots B_m x_m y_m', y_0 z_1).$$

Далее рассуждения с использованием пп.2, 3 построений, а также переходов (1.4) для $i = 2, 3, \dots, m$ повторяются. В результате получаем последующие движения:

$$(q, x_1 t_2 x_2 \dots t_m x_m, x_1 y_1' B_2 x_2 y_2' \dots B_m x_m y_m', y_0 z_1) \overset{*}{\underset{P}{\vdash}}$$
$$\overset{*}{\underset{P}{\vdash}} (q, t_2 x_2 \dots t_m x_m, y_1' B_2 x_2 y_2' \dots B_m x_m y_m', y_0 z_1) \overset{*}{\underset{P}{\vdash}}$$
$$\overset{*}{\underset{P}{\vdash}} (q, t_2 x_2 \dots t_m x_m, B_2 x_2 y_2' \dots B_m x_m y_m', y_0 z_1 y_1) \overset{*}{\underset{P}{\vdash}} \dots$$
$$\overset{*}{\underset{P}{\vdash}} (q, t_m x_m, B_m x_m y_m', y_0 z_1 y_1 \dots z_{m-1} y_{m-1}) \overset{*}{\underset{P}{\vdash}}$$
$$\overset{*}{\underset{P}{\vdash}} (q, x_m, x_m y_m', y_0 z_1 y_1 \dots z_{m-1} y_{m-1} z_m) \overset{*}{\underset{P}{\vdash}}$$
$$\overset{*}{\underset{P}{\vdash}} (q, \varepsilon, y_m', y_0 z_1 y_1 \dots z_{m-1} y_{m-1} z_m) \overset{*}{\underset{P}{\vdash}}$$

$\vdash_P^* (q, \varepsilon, \varepsilon, y_0 z_1 y_1 \ldots z_{m-1} y_{m-1} z_m y_m) = (q, \varepsilon, \varepsilon, y)$.

В конце принято во внимание представление цепочки y согласно равенству (1.1).

Итак, вся последовательность движений может быть записана в виде

$$(q, x, A, \varepsilon) \vdash_P^* (q, \varepsilon, \varepsilon, y).$$

В частности, из доказанного вспомогательного утверждения при $A = S$ следует утверждение I.

II. Докажем теперь обратное утверждение:

$$\text{если } (q, x, S, \varepsilon) \vdash_P^* (q, \varepsilon, \varepsilon, y) \text{ , то } (S, S) \Rightarrow_T^* (x, y).$$

Для этого индукцией по числу l движений типа 1, определенных согласно п.1 построений, докажем более общее утверждение:

если для любого $A \in N$ существует переход

$$(q, x, A, \varepsilon) \vdash_P^* (q, \varepsilon, \varepsilon, y),$$

то

$$(A, A) \Rightarrow_T^* (x, y).$$

База. Пусть $l = 1$. Имеем $(q, x, A, \varepsilon) \vdash_P^* (q, \varepsilon, \varepsilon, y)$, где только одно движение типа 1. Очевидно, что оно — первое движение, так как в исходной конфигурации на вершине магазина находится $A \in N$. Это движение не может привести к появлению нетерминалов в магазинной цепочке из-за того, что они неизбежно привели бы к другим движениям типа 1. Кроме того, магазинная цепочка, замещающая A на вершине магазина, должна начинаться на x, так как только в этом случае удастся продвинуться по входу x (в результате движений, определенных в п.2). Наконец, магазинная цепочка должна заканчиваться на y', потому что только в этом случае на выходе может образоваться цепочка y (в результате движений, определенных в п.3). Поэтому фактически имеем

$$(q, x, A, \varepsilon) \vdash_P (q, x, xy', \varepsilon) \vdash_P^* (q, \varepsilon, y', \varepsilon) \vdash_P^* (q, \varepsilon, \varepsilon, y),$$

где первое движение обусловлено тем, что $(q, xy', \varepsilon) \in \delta(q, \varepsilon, A)$, а это означает существование правила $A \to x, y \in R$. Два последних перехода выполнены согласно пп.2, 3 построений. Воспользовавшись существующим правилом, немедленно получаем вывод $(A, A) \Rightarrow_T (x, y)$.

Индукционная гипотеза. Предположим, что вспомогательное утверждение выполняется для всех $l \le n$ $(n \ge 1)$.

Индукционный переход. Докажем утверждение для $l = n + 1$. Пусть имеется переход $(q, x, A, \varepsilon) \vdash_P^* (q, \varepsilon, \varepsilon, y)$, в котором ровно $n + 1$ движение типа 1. Поскольку в исходной конфигурации на вершине магазина $A \in N$, то первое же движение — типа 1:

$$(q, x, A, \varepsilon) \vdash_P (q, x, x_0 y_0' B_1 x_1 y_1' \ldots B_m x_m y_m', \varepsilon) \vdash_P^* (q, \varepsilon, \varepsilon, y). \quad (1.5)$$

В конечной конфигурации магазин пуст. Цепочка $x_0 \in \Sigma^*$, появившаяся в верхней части магазина после первого движения, может быть удалена, только

если входная цепочка x начинается на x_0. Поэтому далее последуют движения, определяемые п.2, которые продвинут вход по x_0 и удалят такую же цепочку из магазина. Далее ряд движений, определяемых п.3, удалит цепочку y_0' из магазина, выдав на выход y_0, и символ B_1 окажется на вершине магазина. К моменту, когда вершина магазина опустится ниже этой позиции, будет просканирована некоторая часть входа t_1, следующая за цепочкой x_0, а на выходе образуется некоторая цепочка z_1:

$(q, x, A, \varepsilon) = (q, x_0 t_1 x', A, \varepsilon) \vdash_P (q,\ x_0 t_1 x', x_0 y_0' B_1 x_1 y_1' \dots B_m x_m y_m', \varepsilon) \vdash_P^*$
$\vdash_P^* (q,\ t_1 x', y_0' B_1 x_1 y_1' \dots B_m x_m y_m',\ \varepsilon) \vdash_P^* (q,\ t_1 x', B_1 x_1 y_1' \dots B_m x_m y_m', y_0) \vdash_P^*$
$\vdash_P^* (q,\ x', x_1 y_1' \dots B_m x_m y_m',\ y_0 z_1).$

Далее мы можем повторить рассуждения, аналогичные рассуждениям предыдущего абзаца, относя их к цепочкам $x_i \in \Sigma^*$, $y_i' \in \Delta'$ $(i = 1, 2, \dots, m)$ и $B_j \in N$ $(j = 2, \dots, m)$, последовательно появляющимся в верхней части магазина в результате серии движений, построенных в соответствии с п.2, затем с п.3, и ряда движений, приводящих к понижению вершины магазина ниже позиции, занимаемой B_j. Другими словами, детальный разбор возможных движений от исходной конфигурации к конечной дает основание утверждать, что вход x представим в виде

$$x = x_0 t_1 x_1 \dots t_m x_m,\ \ y = x_0 t_1 x_1 \dots t_m x_m, \tag{1.6}$$

причем

$$(q, t_i, B_i, \varepsilon) \vdash_P^{l_i} (q, \varepsilon, \varepsilon, z_i),\ l_i \le n,\ i = 1, 2, \dots, m. \tag{1.7}$$

По построению первое движение (1.4) обусловлено существованием правила

$$A \to x_0 B_1 x_1 \dots B_m x_m, y_0 B_1 y_1 \dots B_m y_m \in R, \tag{1.8}$$

а из существования движений (1.7) по индукционному предположению следует существование выводов

$$(B_i, B_i) \overset{*}{\underset{T}{\Rightarrow}} (t_i, z_i),\ i = 1, 2, \dots, m. \tag{1.9}$$

Используя движения (1.8) и (1.9), с учетом (1.6) получаем:

$(A, A) \underset{T}{\Rightarrow} (x_0 B_1 x_1 \dots B_m x_m, y_0 B_1 y_1 \dots B_m y_m) \overset{*}{\underset{T}{\Rightarrow}} (x_0 t_1 x_1 \dots t_m x_m, y_0 z_1 y_1 \dots z_m y_m) = (x, y).$

В частности, при $A = S$ следует утверждение II.

Из рассуждений I и II следует утверждение леммы.

Доказанная лемма дает алгоритм построения недетерминированного магазинного преобразователя, эквивалентного данной простой схеме синтаксически управляемой трансляции.

Пример 1.4. Пусть sdts $T = (\{E\}, \{a, +, *\}, \{a, +, *\}, R, E)$, где

$R = \{(1)\ E \to +EE, EE+\ ;\ (2)\ E \to *EE, EE*\ ;\ (3)\ E \to a, a\}.$

По лемме 10.1 эквивалентный магазинный преобразователь есть

$P = (\{q\}, \{a, +, *\}, \{E, a, +, *, a', +', *'\}, \{a, +, *\}, \delta, q, E, \varnothing)$, где

1) $\delta(q, \varepsilon, E) = \{(q, +EE+', \varepsilon), (q, *EE*', \varepsilon), (q, aa', \varepsilon)\}$,

2) $\delta(q, b, b) = \{(q, \varepsilon, \varepsilon)\}$ для всех $b \in \{a, +, *\}$,

3) $\delta(q, \varepsilon, c') = \{(q, \varepsilon, c)\}$ для всех $c \in \{a, +, *\}$.

Сравните этот недетерминированный магазинный преобразователь с эквивалентным детерминированным pdt из примера 1.3. Оба преобразуют префиксные арифметические выражения в постфиксные.

Лемма 1.2. *Пусть $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, \varnothing)$ — недетерминированный магазинный преобразователь. Существует простая схема синтаксически-управляемой трансляции T, такая, что $\tau(T) = \tau_e(P)$.*

Доказательство. Построим такую схему T следующим образом. Положим $T = (N, \Sigma, \Delta, R, S)$, где $N = \{S\} \cup \{[qZp] \mid q, p \in Q, Z \in \Gamma\}$, Σ и Δ — такие же, как в pdt P,

$R = \{S \to [q_0 Z_0 p], [q_0 Z_0 p] \mid$ для всех $p \in Q\} \cup$

$\cup \{[qZp] \to a[q_1 Z_1 q_2][q_2 Z_2 q_3] \ldots [q_m Z_m q_{m+1}], y[q_1 Z_1 q_2][q_2 Z_2 q_3] \ldots [q_m Z_m q_{m+1}] \mid$ $(q_1, Z_1 Z_2 \ldots Z_m, y) \in \delta(q, a, Z)$; $a \in \Sigma \cup \{\varepsilon\}$, $y \in \Delta^*$; $p, q, q_i \in Q$; $Z, Z_i \in \Gamma$;

$i = 1, 2, \ldots, m+1$; $q_{m+1} = p\}$.

I. Докажем сначала, что если $(q, x, Z, \varepsilon) \overset{*}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon, y)$, то $([qZp], [qZp]) \overset{*}{\underset{T}{\Rightarrow}} (x, y)$, используя индукцию по числу l движений магазинного преобразователя P.

База. Пусть $l = 1$. Имеем $(q, x, Z, \varepsilon) \underset{P}{\vdash} (p, \varepsilon, \varepsilon, y)$. В этом случае $x \in \Sigma \cup \{\varepsilon\}$ и $(p, \varepsilon, y) \in \delta(q, x, Z)$. Тогда по построению схемы T существует правило $[qZp] \to x, y \in R$, с помощью которого немедленно получаем: $([qZp], [qZp]) \underset{T}{\Rightarrow} (x, y)$.

Индукционная гипотеза. Предположим, что утверждение I выполняется для всех переходов между конфигурациями за число движений $l \le n$ $(n \le 1)$.

Индукционный переход. Докажем, что тогда утверждение I справедливо и для $l = n + 1$. Итак, пусть $(q, x, Z, \varepsilon) \overset{n+1}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon, y)$. Рассмотрим этот переход подробнее.

В общем случае первое движение имеет вид

$$(q, x, Z, \varepsilon) = (q, ax', Z, \varepsilon) \underset{P}{\vdash} (q_1, x', Z_1 Z_2 \ldots Z_m, y_0). \tag{1.10}$$

Затем следуют дальнейшие движения:

$(q_1, x', Z_1 Z_2 \ldots Z_m, y_0) = (q_1, x_1 x_2 \ldots x_m, Z_1 \ldots Z_m, y_0) \overset{l_1}{\underset{P}{\vdash}} (q_2, x_2 \ldots x_m, Z_2 \ldots Z_m, y_0 y_1) \overset{l_2}{\underset{P}{\vdash}} \ldots$

$\ldots \overset{l_m}{\underset{P}{\vdash}} (q_{m+1}, \varepsilon, \varepsilon, y_0 y_1 y_2 \ldots y_m) = (p, \varepsilon, \varepsilon, y)$.

Здесь $a \in \Sigma \cup \{\varepsilon\}$, $x = a x_1 x_2 \ldots x_m$, $y = y_0 y_1 y_2 \ldots y_m$, причем

$$(q_i, x_i, Z_i, \varepsilon) \overset{l_i}{\underset{P}{\vdash}} (q_{i+1}, \varepsilon, \varepsilon, y_i),\ i = 1, 2, \ldots, m;\ l_i \le n;\ q_{m+1} = p. \tag{1.11}$$

Первое движение (1.10) существует потому, что $(q_1, Z_1 \ldots Z_m, y_0) \in \delta(q, a, Z)$, следовательно, по способу построения правил схемы в ней имеется правило

$$[qZp] \to a[q_1 Z_1 q_2][q_2 Z_2 q_3] \ldots [q_m Z_m q_{m+1}],\ y_0[q_1 Z_1 q_2][q_2 Z_2 q_3] \ldots [q_m Z_m q_{m+1}], \tag{1.12}$$

в обозначениях нетерминалов которого участвуют те состояния, по которым проходил pdt P. Из последующих движений (1.12) согласно индукционной гипотезе следует существование выводов

$$([q_i Z_i q_{i+1}], [q_i Z_i q_{i+1}]) \overset{*}{\underset{T}{\Rightarrow}} (x_i, y_i),\ i = 1, 2, \ldots, m. \tag{1.13}$$

Из (1.12) и (1.13) можно выстроить требуемый вывод:

$([qZp], [qZp]) \underset{T}{\Rightarrow} (a[q_1 Z_1 q_2] \ldots [q_m Z_m q_{m+1}], y_0[q_1 Z_1 q_2] \ldots [q_m Z_m q_{m+1}]) \overset{*}{\underset{T}{\Rightarrow}}$

$\overset{*}{\underset{T}{\Rightarrow}} (ax_1x_2\ldots x_m, y_0y_1\ldots y_m) = (x, y)$.

II. Индукцией по длине l вывода докажем теперь обратное утверждение: если $([qZp], [qZp]) \overset{*}{\underset{T}{\Rightarrow}} (x, y)$, то $(q, x, Z, \varepsilon) \overset{*}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon, y)$.

База. Пусть $l = 1$. Имеем $([qZp], [qZp]) \underset{T}{\Rightarrow} (x, y)$. На единственном шаге этого вывода использовано правило схемы $[qZp] \to x, y$, которое обязано своим происхождением тому, что $(p, \varepsilon, y) \in \delta(q, x, Z)$, где $x \in \Sigma \cup \{\varepsilon\}$, $y \in \Delta^*$, а тогда $(q, x, Z, \varepsilon) \underset{P}{\vdash} (p, \varepsilon, \varepsilon, y)$.

Индукционная гипотеза. Предположим, что аналогичное утверждение выполняется для всех выводов, длина которых не превосходит n $(0 \le n \le 1)$.

Индукционный переход. Докажем аналогичное утверждение для выводов длиной $l = n + 1$. Пусть

$$([qZp], [qZp]) \underset{T}{\Rightarrow} (a[q_1Z_1q_2]\ldots[q_mZ_mq_{m+1}], y_0[q_1Z_1q_2]\ldots[q_mZ_mq_{m+1}]) \overset{n}{\underset{T}{\Rightarrow}} (x, y). \quad (1.14)$$

Из (1.14) следует, что существует правило

$$[qZp] \to a[q_1Z_1q_2]\ldots[q_mZ_mq_{m+1}],\ y[q_1Z_1q_2]\ldots[q_mZ_mq_{m+1}] \in R,$$

которое обязано своим происхождение тому, что

$$(q_1, Z_1\ldots Z_m,\ y) \in \delta(q, a, Z). \quad (1.15)$$

Кроме того, из (1.14) следует, что

$$x = ax_1\ldots x_m,\ \ y = y_0y_1\ldots y_m, \quad (1.16)$$

$$([q_iZ_iq_{i+1}],[q_iZ_iq_{i+1}]) \overset{l_i}{\underset{T}{\Rightarrow}} (x_i, y_i),\ l_i \le n,$$

а тогда согласно индукционной гипотезе

$$(q_i, x_i, Z_i, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q_{i+1}, \varepsilon, \varepsilon, y_i),\ i = 1, 2, \ldots, m;\ q_{m+1} = p. \quad (1.17)$$

Из (1.15) и (1.17) с учетом (1.16) выстраивается последовательность движений:

$$(q, x, Z, \varepsilon) = (q,\ ax_1\ldots x_m, Z, \varepsilon) \underset{P}{\vdash} (q_1, x_1\ldots x_m, Z_1\ldots Z_m, y_0) \overset{*}{\underset{P}{\vdash}}$$
$$\overset{*}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon, y_0y_1\ldots y_m) = (p, \varepsilon, \varepsilon, y).$$

Из рассуждений I и II следует, что для любых $q, p \in Q$ и $Z \in \Gamma$ вывод

$$([qZp], [qZp]) \overset{*}{\underset{T}{\Rightarrow}} (x, y)$$

существует тогда и только тогда, когда

$$(q, x, Z, \varepsilon) \overset{*}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon, y).$$

В частности, это справедливо для $Z = Z_0$.

В то же время при помощи правила вида $S \to [q_0Z_0p]$, $[q_0Z_0p]$ всегда можно пристроить начало к вышеприведенному выводу, чтобы считать доказанным утверждение

$$(S, S) \underset{T}{\Rightarrow} ([q_0Z_0p], [q_0Z_0p]) \overset{*}{\underset{T}{\Rightarrow}} (x, y)$$

тогда и только тогда, когда

$$(q, x, Z_0, \varepsilon) \overset{*}{\underset{P}{\vdash}} (p, \varepsilon, \varepsilon, y).$$

Лемма доказана.

Из лемм 1.1 и 1.2 следует

Теорема 1.1. *Трансляция* $\tau = \tau(T)$, *где* T *— простая схема синтаксически управляемой трансляции тогда и только тогда, когда существует недетерминированный магазинный преобразователь* P, *такой, что* $\tau_e(P) = \tau$.

Пример 1.5. В предыдущем примере по простой sdts $T = (\{E\}, \{a, +, *\}, \{a, +, *\}, R, E)$, где $R = \{(1)\ E \to +EE,\ EE+\ ;\ (2)\ E \to *EE,\ EE*\ ;\ (3)\ E \to a,\ a\}$, был построен эквивалентный магазинный преобразователь

$P = (\{q\}, \{a, +, *\}, \{E, a, +, *, a', +', *'\}, \{a, +, *\}, \delta, q, E, \varnothing)$,

где

1) $\delta(q, \varepsilon, E) = \{(q, +EE+', \varepsilon), (q, *EE*', \varepsilon), (q, aa', \varepsilon)\}$,

2) $\delta(q, b, b) = \{(q, \varepsilon, \varepsilon)\}$ для всех $b \in \{a, +, *\}$,

3) $\delta(q, \varepsilon, c') = \{(q, \varepsilon, c)\}$ для всех $c \in \{a, +, *\}$.

Теперь по этому недетерминированному преобразователю P мы построим эквивалентную простую схему синтаксически управляемой трансляции, воспользовавшись алгоритмом, описанным в лемме 1.2. Положим

$T = (\{S, [qEq], [q+q], [q*q]\}, \{a, +, *\}, \{a, +, *\}, R, S)$,

$R = \{$(1) $S \to [qEq], [qEq]$;

(2) $[qEq] \to [q+q]\ [qEq]\ [qEq]\ [q+'q], [q+q]\ [qEq]\ [qEq]\ [q+'q]$;

(3) $[qEq] \to [q*q]\ [qEq]\ [qEq]\ [q*'q], [q*q]\ [qEq]\ [qEq]\ [q*'q]$;

(4) $[qEq] \to [qaq]\ [qa'q], [qaq]\ [qa'q]$;

(5) $[qaq] \to a, \varepsilon$;

(6) $[q+q] \to +, \varepsilon$;

(7) $[q*q] \to *, \varepsilon$;

(8) $[qa'q] \to \varepsilon, a$;

(9) $[q+'q] \to \varepsilon, +$;

(10) $[q*'q] \to \varepsilon, *\}$.

Эта схема мало похожа на исходную, в которой было всего три правила. Однако ее можно эквивалентными преобразованиями привести к исходной.

Во-первых, правые части правил 5–10 можно подставить в правые части правил 2–4. В результате получим

$R' = \{$(1) $S \to [qEq], [qEq]$;

(2′) $[qEq] \to +[qEq]\ [qEq], [qEq]\ [qEq]+$;

(3′) $[qEq] \to *[qEq]\ [qEq], [qEq]\ [qEq]*$;

(4′) $[qEq] \to a, a\}$.

Легко видеть, что из (S, S) выводится в точности то же, что и из $([qEq], [qEq])$. Остается заменить в правилах 2′–4′ слева и справа $[qEq]$ на простое E и отбросить бесполезное правило 1, чтобы получить исходную схему.

Лемма 1.3. *Пусть* $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F)$ *— недетерминированный магазинный преобразователь и* $\tau = \tau(P)$. *Существует недетерминированный магазинный преобразователь* P', *такой, что* $\tau_e(P') = \tau$.

Доказательство. Построим pdt P', исходя из того соображения, что он будет моделировать pdt P до тех пор, пока тот не примет свою входную цепочку, а затем pdt P' будет опустошать свой магазин, совершая ε-движения и не выдавая ничего на выход. Таким образом, pdt P' будет принимать то же самое множество входных цепочек при пустом магазине, какое pdt P принимает при конечных состояниях, и выдавать на выход такие же выходные цепочки.

Итак, положим $P' = (Q', \Sigma, \Gamma', \Delta, \delta', q_0', Z_0', \varnothing)$, где входной и выходной алфавиты — такие же, как у pdt P, а множество конечных состояний, которое в этом случае несущественно, пусто; $Q' = Q \cup \{q_0', q_e\}$, $q_0', q_e \notin Q$; $\Gamma' = \Gamma \cup \{Z_0'\}$, $Z_0' \notin \Gamma$.

Отображение δ' определяется следующим образом:

1. $\delta'(q_0', \varepsilon, Z_0') = \{(q_0, Z_0 Z_0', \varepsilon)\}$ воспроизводит начальную конфигурацию недетерминированного магазинного преобразователя P.

2. $\delta'(q, a, Z)$ содержит все элементы $\delta(q, a, Z)$ для $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$, — реализуется собственно моделирование движений pdt P.

3. $\delta'(q, \varepsilon, Z)$ содержит $(q_e, \varepsilon, \varepsilon)$, если $q \in F$, $Z \in \Gamma'$, — происходит переход в состояние-“ловушку”.

4. $\delta'(q_e, \varepsilon, Z) = \{(q_e, \varepsilon, \varepsilon)\}$ для всех $Z \in \Gamma'$ — производится опустошение магазина.

I. Докажем сначала, что если $(x, y) \in \tau(P)$, то $(x, y) \in \tau_e(P')$.

Пусть $(x, y) \in \tau(P)$, т.е. $(q_0, x, Z_0, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \alpha, y)$, где $q \in F$. Посмотрим, как будет действовать pdt P' на таком же входе.

Согласно п.1 построения имеем $(q_0', x, Z_0', \varepsilon) \underset{P'}{\vdash} (q_0, x, Z_0 Z_0', \varepsilon)$. Далее согласно п.2 pdt P' повторяет все движения pdt P, т.е. $(q_0, x, Z_0 Z_0', \varepsilon) \overset{*}{\underset{P'}{\vdash}} (q, \varepsilon, \alpha Z_0', y)$, где $q \in F$, а тогда согласно п.2 pdt P' переходит в состояние q_e, оставаясь в котором, в соответствии с п.3 опустошает свой магазин: $(q, \varepsilon, \alpha Z_0', y) \overset{*}{\underset{P'}{\vdash}} (q_e, \varepsilon, \varepsilon, y)$. Следовательно, $(x, y) \in \tau(P')$.

II. Докажем теперь обратное, т.е. если $(x, y) \in \tau_e(P')$, то $(x, y) \in \tau(P)$. Пусть $(x, y) \in \tau(P')$, т.е. $(q_0', x, Z_0', \varepsilon) \overset{*}{\underset{P'}{\vdash}} (q_e, \varepsilon, \varepsilon, y)$. Рассмотрим подробнее его движения. Первое движение согласно п.1 имеет вид

$$(q_0', x, Z_0', \varepsilon) \underset{P'}{\vdash} (q_0, x, Z_0 Z_0', \varepsilon).$$

Ясно, что для опустошения магазина pdt P' должен оказаться в состоянии q_e: без этого ему не сбросить символ Z_0', находящийся на “дне” магазина. В свою

очередь, согласно п.2 pdt P' достигает состояния q_e только после того, как, повторяя движения pdt P согласно п.3, он оказывается в состоянии $q \in F$. С этого момента он не продвигается по входной цепочке, ничего не пишет на выходную ленту и только стирает магазин. Это значит, что к моменту достижения конечного состояния он уже прочитал всю свою входную цепочку x и записал всю свою выходную цепочку y.

Дальнейшие движения pdt P' имеют вид

$(q_0, x, Z_0Z_0', \varepsilon) \overset{*}{\underset{P'}{\vdash}} (q, \varepsilon, \alpha Z_0', y) \overset{*}{\underset{P'}{\vdash}} (q_e, \varepsilon, \varepsilon, y)$, где $q \in F$, $\alpha \in \Gamma^*$.

Но на участке $(q_0, x, Z_0Z_0', \varepsilon) \overset{*}{\underset{P'}{\vdash}} (q, \varepsilon, \alpha Z_0', y)$ pdt P' лишь повторяет движения недетерминированного магазинного преобразователя P:

$$(q_0, x, Z_0, \varepsilon) \overset{*}{\underset{P'}{\vdash}} (q, \varepsilon, \alpha, y),\ q \in F,\ \alpha \in \Gamma^*.$$

Следовательно, $(x, y) \in \tau(P)$.

Из рассуждений I и II следует справедливость леммы.

Лемма 1.4. *Пусть $P = (Q, \Sigma, \Gamma, \Delta, \delta, q_0, Z_0, F)$ — недетерминированный магазинный преобразователь и $\tau = \tau_e(P)$. Существует недетерминированный магазинный преобразователь P', такой, что $\tau(P') = \tau$.*

Доказательство. Построим pdt P', исходя из того соображения, что pdt P' будет моделировать pdt P до тех пор, пока тот не примет свою входную цепочку, опустошив магазин, а затем pdt P' совершит еще одно — ε-движение, не записывающее ничего на выходную ленту, и перейдет в свое конечное состояние, принимая ту же входную цепочку. Разумеется, надо позаботиться о том, чтобы к этому моменту магазин не был пуст, иначе никакое дальнейшее движение не будет возможно. Таким образом, pdt P' будет принимать то же самое множество входных цепочек при конечном состоянии, выдавая те же выходные цепочки, что и pdt P.

Положим pdt $P' = (Q', \Sigma, \Gamma', \Delta, \delta', q_0', Z_0', F')$. Здесь $Q' = Q \cup \{q_0', q_f\}$, $q_0', q_f \notin Q$; входной и выходной алфавиты – такие же, как у pdt P; $\Gamma' = \Gamma \cup \{Z_0'\}$, $Z_0' \notin \Gamma$; $F' = \{q_f\}$.

Отображение δ' определяется следующим образом:

1. $\delta'(q_0', \varepsilon, Z_0') = \{(q_0, Z_0Z_0', \varepsilon)\}$ воспроизводит начальную конфигурацию недетерминированного магазинного преобразователя P.

2. $\delta'(q, a, Z)$ содержит все элементы $\delta(q, a, Z)$ для $q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$, — происходит собственно моделирование движений pdt P.

3. $\delta'(q, \varepsilon, Z_0') = \{(q_f, \alpha, \varepsilon)\}$, где $\alpha \in \Gamma^*$ — произвольная магазинная цепочка, — определяет переход в конечное состояние.

I. Докажем сначала, что если $(x, y) \in \tau_e(P)$, то $(x, y) \in \tau(P')$. Пусть $(x, y) \in \tau_e(P)$, т.е. $(q_0, x, Z_0, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$. Посмотрим, как будет действовать pdt P' с такой же входной цепочкой.

Соглсно п.1 $(q_0', x, Z_0', \varepsilon) \vdash_{P'} (q_0, x, Z_0Z_0', \varepsilon)$. Далее согласно п.2 pdt P' повторяет все движения pdt P, т.е. $(q_0, x, Z_0Z_0', \varepsilon) \vdash_{P'}^{*} (q, \varepsilon, Z_0', y)$, а затем согласно п.3 pdt P' переходит в состояние q_f: $(q, \varepsilon, Z_0', y) \vdash_{P'}^{*} (q_f, \varepsilon, \alpha, y)$. Следовательно, $(x, y) \in \tau(P')$.

II. Докажем теперь обратное, т.е. если $(x, y) \in \tau(P')$, то $(x, y) \in \tau_e(P)$. Пусть $(x, y) \in \tau(P')$, т.е. $(q_0', x, Z_0', \varepsilon) \vdash_{P'}^{*} (q_f, \varepsilon, \alpha, y)$ при некотором $\alpha \in \Gamma^*$. Рассмотрим подробнее его движения. Первое движение согласно п.1 имеет вид

$(q_0', x, Z_0', \varepsilon) \vdash_{P'} (q_0, x, Z_0Z_0', \varepsilon)$.

Далее pdt P' оказывается в своем конечном состоянии q_f. Это возможно только в результате движения, построенного в соответствии с п.3, который применим лишь тогда, когда на вершине магазина pdt P' покажется символ Z_0', т.е. дальнейшие движения имеют вид

$(q_0, x, Z_0Z_0', \varepsilon) \vdash_{P'}^{*} (q, \varepsilon, Z_0', y) \vdash_{P'} (q_f, \varepsilon, \alpha, y)$.

На участке $(q_0, x, Z_0Z_0', \varepsilon) \vdash_{P'}^{*} (q, \varepsilon, Z_0', y)$ магазинный преобразователь P' просто повторяет движения pdt P: $(q_0, x, Z_0, \varepsilon) \vdash_{P}^{*} (q, \varepsilon, \varepsilon, y)$. А это значит, что $(x, y) \in \tau_e(P)$.

Из рассуждений I и II следует утверждение леммы.

Из лемм 1.3 и 1.4 следует

Теорема 1.2. *Классы трансляций, реализуемых недетерминированными магазинными преобразователями при конечном состоянии и пустом магазине, совпадают.*

Замечание 1.1. Принимая во внимание теорему 1.2, формулировку теоремы 1.1 можно усилить, не подчеркивая того факта, что простая трансляция реализуется недетерминированным магазинным преобразователем при пустом магазине.

К сожалению, не всякая, даже простая, синтаксически управляемая трансляция может быть реализована *детерминированным* МП-преобразователем.

§ 1.4. Детерминированная генерация выходной цепочки трансляции по левостороннему анализу входной цепочки

***Определение* 1.11.** Схема синтаксически управляемой трансляции называется *семантически однозначной*, если в ней не существует двух правил вида $A \to \alpha, \beta$ и $A \to \alpha, \gamma$, в которых $\beta \neq \gamma$.

Другими словами, семантическая цепочка однозначно определяется правилом входной грамматики.

***Определение* 1.12.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика, правила которой занумерованы как $1, 2, \ldots, p$ и $S \underset{lm}{\overset{\pi}{\Rightarrow}} x$ — левосторонний вывод $x \in V_T^*$ в грамматике G. Последовательность номеров правил π, примененных в этом выводе, называется *левосторонним анализом цепочки* x.

Теорема 1.3. *Пусть* $T = (N, \Sigma, \Delta, R, S)$ — *простая семантически однозначная схема синтаксически управляемой трансляции, правила которой занумерованы как* $1, 2, \ldots, p$. *Существует детерминированный магазинный преобразователь* P, *такой, что* $\tau_e(P) = \{(\pi, y) \mid (S, S) \underset{\text{lm}}{\overset{\pi}{\Rightarrow}} (x, y)$ *для некоторой цепочки* $x \in \Sigma^*\}$.

Говоря неформально, выходная цепочка трансляции y может быть сгенерирована детерминированным магазинным преобразователем (dpdt) по левостороннему анализу π входной цепочки x.

Доказательство. Построим dpdt P, о котором идет речь, положив

$$P = (\{q\}, \{1, 2, \ldots, p\}, N \cup \Delta, \Delta, \delta, q, S, \varnothing),$$

где δ определяется следующим образом:

1. $\delta(q, i, A) = (q, \beta, \varepsilon)$, если $A \to \alpha, \beta$ — i-е правило схемы, единственное, начинающееся на $A \to \alpha$.

2. $\delta(q, \varepsilon, b) = (q, \varepsilon, b)$ для всех $b \in \Delta$.

Магазинный преобразователь P — детерминирован, так как правила вида 1 применимы, только если на вершине магазина находится нетерминал, а правила вида 2 используются только тогда, когда на вершине магазина находится выходной символ. Остается доказать, что построенный dpdt P действительно реализует требуемую трансляцию.

I. Индукцией по длине вывода l покажем, что если для любого $A \in N$ существует левосторонний вывод $(A, A) \underset{\text{lm}}{\overset{\pi}{\Rightarrow}} (x, y)$, то $(q, \pi, A, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$.

База. Пусть $l = 1$. На единственном шаге вывода $(A, A) \underset{\text{lm}}{\overset{i}{\Rightarrow}} (x, y)$ применяется правило схемы $A \to x, y$ с номером i. В соответствии с п.1 построения $\delta(q, i, A) = (q, y, \varepsilon)$, и тогда $(q, i, A, \varepsilon) \underset{P}{\vdash} (q, \varepsilon, y, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$. Последний переход совершен согласно п.2 построений, так как $y \in \Delta^*$.

Индукционная гипотеза. Предположим, что подобное утверждение выполняется для всех выводов длиной $l \le n$ $(n \ge 1)$.

Индукционный переход. Покажем, что утверждение выполняется и для $l = n + 1$.

Пусть $(A, A) \underset{\text{lm}}{\overset{i}{\Rightarrow}} (x_0 A_1 x_1 A_2 \ldots A_m x_m,\ y_0 A_1 y_1 A_2 \ldots A_m y_m) \underset{\text{lm}}{\overset{\pi'}{\Rightarrow}} (x, y)$ — вывод длиной $n + 1$; $|\pi'| = n$; $A, A_j \in N$; $x_j \in \Sigma^*$, $y_j \in \Delta^*$, $j = 1, 2, \ldots, m$. На первом шаге применяется правило схемы $A \to x_0 A_1 x_1 A_2 \ldots A_m x_m, y_0 A_1 y_1 A_2 \ldots A_m y_m$, имеющее номер i. Согласно п.1

$$\delta(q, i, A) = (q, y_0 A_1 y_1 A_2 \ldots A_m y_m, \varepsilon). \tag{1.18}$$

Одновременно ясно, что

$$x = x_0 t_1 x_1 t_2 \ldots t_m x_m, \quad y = y_0 z_1 y_1 z_2 \ldots z_m y_m, \quad \pi = i \pi_1 \pi_2 \ldots \pi_m, \tag{1.19}$$

где $(A_j, A_j) \underset{\text{lm}}{\overset{\pi_j}{\Rightarrow}} (t_j, z_j)$, $|\pi_j| \le n$, и, следовательно, по индукционной гипотезе

$$(q, \pi_j, A_j, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, z_j) \text{ для } j = 1, 2, \ldots, m. \tag{1.20}$$

Принимая во внимание (1.18)–(1.20), а также п.2, можем написать:

$$(q, \pi, A, \varepsilon) = (q, i\pi_1\pi_2 \ldots \pi_m, A, \varepsilon) \underset{P}{\vdash} (q, \pi_1\pi_2 \ldots \pi_m, y_0 A_1 y_1 A_2 \ldots A_m y_m, \varepsilon) \overset{*}{\underset{P}{\vdash}}$$
$$\overset{*}{\underset{P}{\vdash}} (q, \pi_1\pi_2 \ldots \pi_m, A_1 y_1 A_2 \ldots A_m y_m, y_0) \overset{*}{\underset{P}{\vdash}} (q, \pi_2 \ldots \pi_m, y_1 A_2 \ldots A_m y_m, y_0 z_1) \overset{*}{\underset{P}{\vdash}} \ldots$$
$$\ldots \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, y_m, y_0 z_1 y_1 \ldots z_m) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y_0 z_1 y_1 \ldots z_m y_m) = (q, \varepsilon, \varepsilon, y).$$

II. Индукцией по длине ввода $l = |\pi|$ покажем, что для любого $A \in N$, если $(q, \pi, A, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$, то $(A, A) \overset{\pi}{\underset{lm}{\Rightarrow}} (x, y)$ при некотором $x \in \Sigma^*$.

База. Пусть $l = 1$, т.е. $\pi = i$, и $(q, i, A, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$. В общем случае этот переход может происходить только следующим образом: $(q, i, A, \varepsilon) \underset{P}{\vdash} (q, \varepsilon, y, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$. Действительно, первое движение происходит согласно п.1, поскольку на вершине магазина $A \in N$. Других движений этого типа не существует, так как каждое из них продвигается по входной цепочке на один символ. Возможные же движения согласно п.2 предполагают нахождение в верхней части магазина некоторой цепочки $y \in \Delta^*$.

Первое движение определяется значением $\delta(q, i, A) = (q, y, \varepsilon)$, предполагающим существование правила вида $A \to x, y \in R$ с номером i при некотором $x \in \Sigma^*$. С его помощью немедленно получаем $(A, A) \overset{i}{\underset{lm}{\Rightarrow}} (x, y)$.

Индукционная гипотеза. Предположим, что подобное утверждение выполняется для всех вводов длиной $l \le n$ $(n \ge 1)$.

Индукционный переход. Предположим, что утверждение выполняется и для $l = n + 1$. Пусть $(q, \pi, A, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y)$, где $|\pi| = n + 1$. В общем случае эти движения могут происходить только следующим образом:

$(q, \pi, A, \varepsilon) = (q, i\pi', A, \varepsilon) \underset{P}{\vdash} (q, \pi', y_0 A_1 y_1 A_2 \dots A_m y_m, \varepsilon) \overset{*}{\underset{P}{\vdash}}$

$\overset{*}{\underset{P}{\vdash}} (q, \pi', A_1 y_1 A_2 \dots A_m y_m, y_0) = (q, \pi_1 \pi_2 \dots \pi_m, A_1 y_1 A_2 \dots A_m y_m, y_0) \overset{*}{\underset{P}{\vdash}}$

$\overset{*}{\underset{P}{\vdash}} (q, \pi_2 \dots \pi_m, y_1 A_2 \dots A_m y_m, y_0 z_1) \overset{*}{\underset{P}{\vdash}} (q, \pi_2 \dots \pi_m, A_2 \dots A_m y_m, y_0 z_1 y_1) \overset{*}{\underset{P}{\vdash}} \dots$

$\dots \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, y_m, y_0 z_1 y_1 \dots z_m) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, y_0 z_1 y_1 \dots z_m y_m) = (q, \varepsilon, \varepsilon, y).$

Соответственно

$$\pi = i\pi_1 \pi_2 \dots \pi_m, \quad y = y_0 z_1 y_1 z_2 \dots z_m y_m. \tag{1.21}$$

Первое движение позволяет утверждать, что $\delta(q, i, A) = (q, y_0 A_1 y_1 A_2 \dots A_m y_m, \varepsilon)$, и, следовательно, в схеме существует правило под номером i вида

$$A \to x_0 A_1 x_1 A_2 \dots A_m x_m, y_0 A_1 y_1 A_2 \dots A_m y_m. \tag{1.22}$$

Промежуточные движения вида $(q, \pi_j, A_j, \varepsilon) \overset{*}{\underset{P}{\vdash}} (q, \varepsilon, \varepsilon, z_j)$, $|\pi_j| \le n$, по индукционному предположению гарантируют существование выводов вида

$$(A_j, A_j) \overset{\pi_j}{\underset{lm}{\Rightarrow}} (t_j, z_j) \text{ при некоторых } t_j \in \Delta^*, \ j = 1, 2, \dots, m. \tag{1.23}$$

Используя правило (1.22), выводы (1.23) и учитывая равенство (1.21), можем построить вывод

$$(A, A) \overset{i}{\underset{lm}{\Rightarrow}} (x_0 A_1 x_1 A_2 \dots A_m x_m, y_0 A_1 y_1 A_2 \dots A_m y_m) \overset{\pi'}{\underset{lm}{\Rightarrow}}$$

$$\overset{\pi'}{\underset{lm}{\Rightarrow}} (x_0 t_1 x_1 t_2 \dots t_m x_m, y_0 z_1 y_1 z_2 \dots z_m y_m) = (x, y).$$

Здесь $\pi' = \pi_1 \pi_2 \dots \pi_m$, $i\pi' = \pi$, т.е. $(A, A) \overset{\pi}{\underset{lm}{\Rightarrow}} (x, y)$.

Утверждение теоремы следует из рассуждений I и II при $A = S$.

Пример 1.6. Пусть $T = (\{E, T, F\}, \{a, +, *, (,)\}, \{a, +, *\}, R, E)$, где

$R = \{$(1) $E \to E + T, +ET$;
(2) $E \to T, T$;
(3) $T \to T * F, *TF$;
(4) $T \to F, F$;
(5) $F \to (E), E$;
(6) $F \to a, a$ $\}$.

В соответствии с описанием построений в теореме 1.3 определим детерминированный магазинный преобразователь, восстанавливающий выход трансляции по левостороннему анализу входной цепочки:

$P = (\{q\}, \{1, 2, 3, 4, 5, 6\}, (\{E, T, F, a, +, *\}, \{a, +, *\}, \delta, q, E, \varnothing)$,

где

1) $\delta(q, 1, E) = (q, +ET, \varepsilon)$;
2) $\delta(q, 2, E) = (q, T, \varepsilon)$;
3) $\delta(q, 3, T) = (q, *TF, \varepsilon)$;
4) $\delta(q, 4, T) = (q, F, \varepsilon)$;
5) $\delta(q, 5, F) = (q, E, \varepsilon)$;
6) $\delta(q, 6, F) = (q, a, \varepsilon)$;
7) $\delta(q, \varepsilon, a) = (q, \varepsilon, a)$;
8) $\delta(q, \varepsilon, +) = (q, \varepsilon, +)$;
9) $\delta(q, \varepsilon, *) = (q, \varepsilon, *)$.

Очевидно, что при $\pi = 23465124646$ имеем $(E, E) \underset{\text{lm}}{\overset{\pi}{\Rightarrow}} (a * (a + a), * a + aa)$. Нетрудно убедиться в том, что $(q, 23465124646, E, \varepsilon) \overset{*}{\underset{\text{p}}{\vdash}} (q, \varepsilon, \varepsilon, * a + aa)$.

Глава 2

LL(*k*)-ГРАММАТИКИ И ТРАНСЛЯЦИИ

§ 2.1. Введение в *LL*(*k*)-грамматики

2.1.1. Неформальное описание

В гл. 1 было показано, что произвольная простая синтаксически управляемая трансляция всегда может быть реализована при помощи недетерминированного магазинного преобразователя. Если же, кроме того, схема, посредством которой задается эта трансляция, является семантически однозначной, то выход трансляции можно получить по левостороннему анализу входной цепочки при помощи детерминированного магазинного преобразователя. Следовательно, если найти такие классы входных грамматик, в которых левосторонний анализ может быть реализован детерминированным образом, то мы имели бы полностью детерминированную реализацию простых семантически однозначных трансляций.

Одним из таких подклассов КС-грамматик являются так называемые *LL*(*k*)-*грамматики*. Это самый большой "естественный" класс левоанализируемых грамматик.

***Определение* 2.1.** Пусть $\alpha = x\beta$ — левосентенциальная форма в некоторой cfg $G = (V_N, V_T, P, S)$, такая, что $x \in V_T^*$, а $\beta \in (V_N \cup V_T)^*$ начинается на нетерминал либо есть пустая цепочка. Цепочка x называется *закрытой частью*, а β — *открытой частью* левосентенциальной формы α.

Пример 2.1. Пусть $\alpha = abacAaB$. Закрытая часть α есть *abac*. Открытая ее часть — *AaB*. Если $\alpha = abc$, то закрытая часть есть *abc*, а открытая — ε.

Пусть $G = (V_N, V_T, P, S)$ — однозначная КС-грамматика и $w = a_1a_2\ldots a_n \in L(G)$. Тогда существует единственная последовательность левосентенциальных форм $\alpha_0, \alpha_1, \ldots, \alpha_m$, такая, что $S = \alpha_0$, $\alpha_i \underset{\text{lm}}{\overset{p_i}{\Rightarrow}} \alpha_{i+1}$ для $0 \le i < m$ и $\alpha_m = w$. Левосторонний анализ цепочки w есть $p_0, p_1, p_2, \ldots, p_{m-1}$.

Теперь предположим, что мы хотим найти этот левосторонний анализ, просматривая w слева направо один раз. Мы могли бы попытаться сделать это путем построения последовательности левосентенциальных форм $\alpha_0, \alpha_1, \ldots, \alpha_m$. Начальная сентенциальная форма $\alpha_0 = S$ уже известна. Остается определить способ построения следующей сентенциальной формы по последней из уже построенных.

Пусть $\alpha_i = a_1a_2\ldots a_jA\beta$ — последняя из построенных сентенциальных форм. Сравнивая закрытую часть этой сентенциальной формы $a_1a_2\ldots a_j$ с цепочкой w, мы можем определить ее окончание $a_{j+1}\ldots a_n$, вывод которого еще предстоит построить. Было бы желательно, чтобы α_{i+1} можно было найти, зная только $a_1a_2\ldots a_j$ — часть входной цепочки, которую мы уже просмотрели к этому мо-

менту, несколько следующих входных символов $a_{j+1}\dots a_{j+k}$ для некоторого фиксированного k и нетерминала A. Если эти три величины однозначно определяют, какое правило должно использоваться для раскрытия A, мы можем тогда точно определить α_{i+1} по α_i и k входным символам $a_{j+1}\dots a_{j+k}$.

Говорят, что грамматика, в которой *каждый* левосторонний вывод имеет это свойство, есть *LL*(k)-*грамматика*.

Будет показано, что для каждой *LL*(k)-грамматики можно построить детерминированный левосторонний анализатор, который действует за линейное время.

2.1.2. Формальное определение

***Определение* 2.2.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. Определим функцию $\mathrm{FIRST}_k^G(\alpha) = \{w \in V_T^* \mid$ либо $|w| < k$ и $\alpha \overset{*}{\underset{G}{\Rightarrow}} w$, либо $|w| = k$ и $\alpha \overset{*}{\underset{G}{\Rightarrow}} wx$ для некоторой цепочки $x \in V_T^*\}$. Здесь $k \geq 0$ — целое, $\alpha \in (V_N \cup V_T)^*$.

Отметим, что эта функция определена, в частности, и для терминальной цепочки, и тогда, когда она пуста. При этом верхний индекс грамматики не существен, так как никакие правила для вывода терминальной цепочки из такого аргумента не требуются.

***Определение* 2.3.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. Говорят, что G есть *LL*(k)-*грамматика* для некоторого фиксированного k, если для любых двух левосторонних выводов вида

1) $S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \overset{*}{\underset{lm}{\Rightarrow}} wx$,

2) $S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{lm}{\Rightarrow}} wy$,

в которых $\mathrm{FIRST}_k^G(x) = \mathrm{FIRST}_k^G(y)$, имеет место равенство $\beta = \gamma$.

***Определение* 2.4.** Говорят, что контекстно-свободная грамматика G есть *LL-грамматика*, если она *LL*(k) для некоторого $k \geq 0$.

Пример 2.2. Пусть $G = (\{S, B\}, \{a, b\}, P, S)$, где $P = \{S \to aBS \mid b, B \to a \mid bSB\}$. По определению грамматика G — *LL*(1), если

1) $S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \overset{*}{\underset{lm}{\Rightarrow}} wx$,

2) $S \overset{*}{\underset{lm}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{lm}{\Rightarrow}} wy$,

и если x и y начинаются на один и тот же символ, то должно быть $\beta = \gamma$.

Рассмотрим два левосторонних вывода, в которых роль нетерминала A играет символ S.

Имеем (1) $S \underset{lm}{\Rightarrow} aBS$ и (2) $S \underset{lm}{\Rightarrow} b$. Тогда $w = \alpha = \varepsilon$, $\beta = aBS$, $\gamma = b$. Ясно, что любая цепочка x, выводимая из $\beta\alpha = aBS$, начинается на a, а цепочка y, выводимая из $\gamma\alpha = b$, равна b. Поэтому если первый символ цепочки, следующей за закрытой частью сентенциальной формы ($w = \varepsilon$), есть a, то для замены нетерминала S следует использовать первую альтернативу. Если она начинается на b, то вторую.

Рассмотрим два левосторонних вывода, в которых роль нетерминала A играет символ B:

Имеем (1) $S \underset{lm}{\Rightarrow} aBS \underset{lm}{\Rightarrow} aaS$ и (2) $S \underset{lm}{\Rightarrow} aBS \underset{lm}{\Rightarrow} abSBS$. Тогда $w = a$, $\alpha = S$, $\beta = a$, $\gamma = bSB$. Ясно, что любая цепочка x, выводимая из $\beta\alpha = aS$, начинается на a, а цепочка y, выводимая из $\gamma\alpha = bSBS$, начинается на b. Поэтому если первый символ цепочки, следующей за закрытой частью сентенциальной формы ($w = a$), есть a, то для замены нетерминала B следует использовать первую альтернативу. Если она начинается на b, то вторую.

В обоих случаях $LL(1)$-условие выполнено: по первому символу цепочки, следующей за закрытой частью сентенциальной формы, однозначно определяется то правило, которое следует применить к соответствующему нетерминалу, чтобы в конце концов получить анализируемую цепочку. Очевидно, что любые два левосторонних вывода в данной грамматике, подпадающие под вышеприведенный образец, удовлетворяют $LL(k)$-условию.

Данная грамматика служит примером *простой* $LL(1)$-грамматики.

***Определение* 2.5.** Говорят, что контекстно-свободная грамматика G является *простой* $LL(1)$-*грамматикой*, если в ней нет ε-правил, и все альтернативы для каждого нетерминала начинаются с терминалов и притом различных.

Таким образом, в простой $LL(1)$-грамматике для данной пары (A, a), где $A \in V_N$ и $a \in V_T$, существует самое большее — одна альтернатива вида $A \to a\alpha$.

§ 2.2. Свойства *LL*(*k*)-грамматик

Теорема 2.1. *Чтобы контекстно-свободная грамматика* $G = (V_N, V_T, P, S)$ *была* $LL(k)$-*грамматикой, необходимо и достаточно, чтобы*

$$\mathrm{FIRST}_k^G(\beta\alpha) \cap \mathrm{FIRST}_k^G(\gamma\alpha) = \varnothing$$

для всех α, β, γ, *таких, что существуют правила* $A \to \beta$, $A \to \gamma \in P$, $\beta \neq \gamma$ *и существует вывод* $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha$.

Доказательство. Будем предполагать, что грамматика G не содержит бесполезных нетерминалов. Это предположение не умаляет общности рассуждений, так как бесполезные нетерминалы не влияют на $LL(k)$-условие, фигурирующее в определении $LL(k)$-грамматик. Обе части доказательства проведем методом “от противного”.

I. *Необходимость*. Пусть G — $LL(k)$-грамматика, но условие не выполнено, т.е. нашлись такие цепочки α, β, γ, что существуют $A \to \beta$, $A \to \gamma \in P$, $\beta \neq \gamma$ и существует вывод $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha$, для которых $\mathrm{FIRST}_k^G(\beta\alpha) \cap \mathrm{FIRST}_k^G(\gamma\alpha) \neq \varnothing$.

Пусть некоторая цепочка $z \in \mathrm{FIRST}_k^G(\beta\alpha) \cap \mathrm{FIRST}_k^G(\gamma\alpha)$. Мы можем продолжить имеющийся вывод $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha$ двумя способами, используя упомянутые два правила и учитывая определение функции FIRST_k^G:

1) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \underset{lm}{\overset{*}{\Rightarrow}} wzu$,

2) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \underset{lm}{\overset{*}{\Rightarrow}} wzv$

для некоторых $u, v \in V_T^*$. При построении этих выводов была использована теорема о том, что всякий вывод терминальной цепочки может быть перестроен в левосторонний.

Остается сопоставить эти два левосторонних вывода с теми, которые участвуют в определении $LL(k)$-грамматик. Здесь в роли цепочки x используется zu, а в роли цепочки y — zv. Очевидно, $\mathrm{FIRST}_k^G(x) = \mathrm{FIRST}_k^G(y) = z$, но $\beta \neq \gamma$, что противоречит предположению о том, что G — $LL(k)$-грамматика.

В частности, если $|z| < k$ и $u = v = \varepsilon$, то имеем два разных левосторонних вывода одной и той же терминальной цепочки wz. Это означает, что G — не однозначная грамматика и, естественно, не $LL(k)$ ни при каком k. Необходимость доказана.

II. *Достаточность*. Пусть условие теоремы выполнено, но G — не $LL(k)$-грамматика. Это значит, что существуют два левосторонних вывода:

1) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \underset{lm}{\overset{*}{\Rightarrow}} wx$,

2) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \underset{lm}{\overset{*}{\Rightarrow}} wy$,

в которых $\mathrm{FIRST}_k^G(x) = \mathrm{FIRST}_k^G(y)$, но $\beta \neq \gamma$.

Пусть $\mathrm{FIRST}_k^G(x) = \mathrm{FIRST}_k^G(y) = z$. Имеем $\beta\alpha \underset{lm}{\overset{*}{\Rightarrow}} x$ и $\gamma\alpha \underset{lm}{\overset{*}{\Rightarrow}} y$. Согласно определению функции FIRST_k^G из существования этих двух выводов заключаем, что

$z = \mathrm{FIRST}_k^G(x) \in \mathrm{FIRST}_k^G(\beta\alpha)$,

$z = \mathrm{FIRST}_k^G(y) \in \mathrm{FIRST}_k^G(\gamma\alpha)$

и, следовательно,

$z \in \mathrm{FIRST}_k^G(\beta\alpha) \cap \mathrm{FIRST}_k^G(\gamma\alpha) \neq \varnothing$,

что входит в противоречие с первоначальным предположением о том, что условие теоремы выполняется. Достаточность доказана, а вместе с ней и вся теорема.

***Определение* 2.6.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика, и $\beta \in V^*$. Определим *функцию* $\mathrm{FOLLOW}_k^G(\beta) = \{w \in V_T^* \mid S \underset{G}{\overset{*}{\Rightarrow}} \gamma\beta\alpha$ и $w \in \mathrm{FIRST}_k^G(\alpha)\}$. Здесь $k \geq 0$ — целое.

Теорема 2.2. *Чтобы контекстно-свободная грамматика* $G = (V_N, V_T, P, S)$ *была* $LL(1)$*-грамматикой, необходимо и достаточно, чтобы*

$$\mathrm{FIRST}_1^G(\beta\,\mathrm{FOLLOW}_1^G(A)) \cap \mathrm{FIRST}_1^G(\gamma\,\mathrm{FOLLOW}_1^G(A)) = \varnothing$$

для всех $A \in V_N$, $\beta, \gamma \in (V_N \cup V_T)^*$, *таких, что существуют правила* $A \to \beta$, $A \to \gamma \in P$, $\beta \neq \gamma$.

Доказательство. Требуется пояснить, что в формулировке условия теоремы аргумент функции FIRST_1^G — не цепочка, как было определено ранее, а множе-

ство цепочек. Расширение области определения этой функции на множества производится естественным образом:

$$\mathrm{FIRST}_1^G(W) = \bigcup_{s \in W} \mathrm{FIRST}_1^G(s),$$

где $W \subseteq V^*$.

Обе части теоремы доказываются методом “от противного”. Как и ранее, не нарушая общности рассуждений, будем считать грамматику G приведенной.

I. *Необходимость*. Пусть G — $LL(1)$-грамматика, но условие не выполнено, т.е. существуют правила $A \to \beta,\ A \to \gamma \in P,\ \beta \neq \gamma,\ c \in (V_\mathrm{T} \cup \{\varepsilon\})$, такие, что

$$c \in \mathrm{FIRST}_1^G(\beta\,\mathrm{FOLLOW}_1^G(A)) \text{ и } c \in \mathrm{FIRST}_1^G(\gamma\,\mathrm{FOLLOW}_1^G(A)).$$

Это означает, что существуют $a, b \in \mathrm{FOLLOW}_1^G(A)$, такие, что $c \in \mathrm{FIRST}_1^G(\beta a)$ и $c \in \mathrm{FIRST}_1^G(\gamma b)$, и согласно определению функции FIRST_1^G существуют выводы:

$$\beta a \overset{*}{\underset{G}{\Rightarrow}} cu \text{ и } \gamma v \overset{*}{\underset{G}{\Rightarrow}} cv, \text{ если } c \in V_\mathrm{T}$$

или

$$\beta a \overset{*}{\underset{G}{\Rightarrow}} \varepsilon \text{ и } \gamma b \overset{*}{\underset{G}{\Rightarrow}} \varepsilon, \text{ если } c = \varepsilon.$$

Случай 1: $c \in V_\mathrm{T}$, $\beta a \overset{*}{\underset{G}{\Rightarrow}} cu\underbrace{a}_{u} = cu$, $\beta \overset{*}{\underset{G}{\Rightarrow}} cu'$, $u' \in V_\mathrm{T}^*$; $\gamma b \overset{*}{\underset{G}{\Rightarrow}} cv'\underbrace{b}_{v}$, $\gamma \overset{*}{\underset{G}{\Rightarrow}} cv'$, $v' \in V_\mathrm{T}^*$.

Так как грамматика G — приведенная, то существует вывод, в частности левосторонний, который порождает сентенциальную форму, содержащую A: $S \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} wA\alpha$, который может быть продолжен двумя способами:

1) $S \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} wA\alpha \underset{\mathrm{lm}}{\Rightarrow} w\beta\alpha \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} wcu'\alpha \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} w\underbrace{cu'z}_{x}$,

2) $S \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} wA\alpha \underset{\mathrm{lm}}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} wcv'\alpha \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} w\underbrace{cv'z}_{y}$,

где вывод $\alpha \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} z$, $z \in V_\mathrm{T}^*$, существует также в силу приведенности грамматики G.

Итак, имеем два левосторонних вывода (1 и 2), фигурирующие в определении $LL(k)$-грамматик, в которых в роли цепочки x используется $cu'z$, в роли цепочки y — $cv'z$, причем

$$\mathrm{FIRST}_1^G(x) = \mathrm{FIRST}_1^G(cu'z) = c,\ \mathrm{FIRST}_1^G(y) = \mathrm{FIRST}_1^G(cv'z) = c,$$

но $\beta \neq \gamma$. Следовательно, грамматика G — не $LL(1)$, что противоречит первоначальному предположению.

Случай 2: $c \in V_\mathrm{T}$, $\beta \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$, $a = c$, $a \in \mathrm{FOLLOW}_1^G(A)$; $\gamma b \overset{*}{\underset{G}{\Rightarrow}} c\underbrace{v'b}_{v}$, $\gamma \overset{*}{\underset{G}{\Rightarrow}} cv'$, $v' \in V_\mathrm{T}^*$.

Согласно определению $c \in \mathrm{FOLLOW}_1^G(A)$ означает, что существует вывод, в частности левосторонний, вида $S \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} wA\alpha$, такой, что $c \in \mathrm{FIRST}_1^G(\alpha)$ или, что то же самое, существует вывод $\alpha \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} cz$ при некотором $z \in V_\mathrm{T}^*$. Продолжим левосторонним образом вывод $S \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} wA\alpha$ двумя способами:

1) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \underset{lm}{\overset{*}{\Rightarrow}} w\alpha \underset{lm}{\overset{*}{\Rightarrow}} w\underbrace{cz}_{x}$,

2) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \underset{lm}{\overset{*}{\Rightarrow}} wcv'\alpha \underset{lm}{\overset{*}{\Rightarrow}} wcv'cz$.

Получили два вывода (1 и 2), фигурирующие в определении $LL(k)$-грамматик, с cz в роли цепочки x и $cv'cz$ в роли цепочки y. Имеем $\mathrm{FIRST}_1^G(x) = \mathrm{FIRST}_1^G(cz) = c$, $\mathrm{FIRST}_1^G(y) = \mathrm{FIRST}_1^G(cv'cz) = c$, т.е. $\mathrm{FIRST}_1^G(x) = \mathrm{FIRST}_1^G(y)$, но $\beta \neq \gamma$. Следовательно, грамматика G — не $LL(1)$, что противоречит первоначальному предположению.

Случай 3: $c \in V_T$, $\beta a \underset{G}{\overset{*}{\Rightarrow}} cu'a$, $\beta \underset{G}{\overset{*}{\Rightarrow}} cu'$, $u' \in V_T^*$; $\gamma \underset{G}{\overset{*}{\Rightarrow}} \varepsilon$, $b = c$, $b \in \mathrm{FOLLOW}_1^G(A)$.

Он разбирается аналогично предыдущему.

Случай 4: $c \in V_T$, $\beta \underset{G}{\overset{*}{\Rightarrow}} \varepsilon$, $\gamma \underset{G}{\overset{*}{\Rightarrow}} \varepsilon$, $a = b = c$, $c \in \mathrm{FOLLOW}_1^G(A)$.

Согласно определению $c \in \mathrm{FOLLOW}_1^G(A)$ означает существование вывода вида $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha$, такого, что $c \in \mathrm{FIRST}_1^G(\alpha)$ или, что то же самое, $\alpha \underset{lm}{\overset{*}{\Rightarrow}} cz$ при некотором $z \in V_T^*$.

Теперь вывод $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha$ продолжим левосторонним образом двумя способами:

1) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \underset{lm}{\overset{*}{\Rightarrow}} w\alpha \underset{lm}{\overset{*}{\Rightarrow}} w\underbrace{cz}_{x}$,

2) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \underset{lm}{\overset{*}{\Rightarrow}} w\alpha \underset{lm}{\overset{*}{\Rightarrow}} w\underbrace{cz}_{y}$,

Мы получили два левосторонних вывода (1 и 2), фигурирующих в определении $LL(k)$-грамматик, в которых в роли цепочек x и y используются одинаковые терминальные цепочки cz, функция FIRST_1^G от них дает одинаковый результат, равный c, притом что $\beta \neq \gamma$. Следовательно, грамматика G — не $LL(1)$, что противоречит первоначальному предположению[12].

Случай 5: $\beta \underset{G}{\overset{*}{\Rightarrow}} \varepsilon$, $\gamma \underset{G}{\overset{*}{\Rightarrow}} \varepsilon$, $a = b = \varepsilon$, $\varepsilon \in \mathrm{FOLLOW}_1^G(A)$.

Как и в случае 4, $\varepsilon \in \mathrm{FOLLOW}_1^G(A)$ означает существование вывода $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha$, такого, что $\alpha \underset{lm}{\overset{*}{\Rightarrow}} \varepsilon$. Его можно продолжить левосторонним образом двумя способами:

1) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \underset{lm}{\overset{*}{\Rightarrow}} w\alpha \underset{lm}{\overset{*}{\Rightarrow}} w$,

2) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \underset{lm}{\overset{*}{\Rightarrow}} w\alpha \underset{lm}{\overset{*}{\Rightarrow}} w$.

[12] Противоречие можно видеть и в том, что существуют два разных левосторонних вывода для одной и той же терминальной цепочки az.

В роли цепочек x и y здесь используется ε. Имеем $\mathrm{FIRST}_1^G(x) = \mathrm{FIRST}_1^G(y) = \varepsilon$, хотя по-прежнему $\beta \neq \gamma$. Следовательно, грамматика G — не $LL(1)$, что противоречит первоначальному предположению[13]. Необходимость доказана.

II. *Достаточность*. Пусть условие теоремы выполнено, но грамматика G — не $LL(1)$. Тогда существуют два левосторонних вывода вида

1) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\beta\alpha \underset{lm}{\overset{*}{\Rightarrow}} wx$,

2) $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\Rightarrow} w\gamma\alpha \underset{lm}{\overset{*}{\Rightarrow}} wy$,

в которых $\mathrm{FIRST}_1^G(x) = \mathrm{FIRST}_1^G(y)$, но $\beta \neq \gamma$. Поскольку $\beta\alpha \underset{lm}{\overset{*}{\Rightarrow}} x$ и $\gamma\alpha \underset{lm}{\overset{*}{\Rightarrow}} y$, то

$$\mathrm{FIRST}_1^G(x) \subseteq \mathrm{FIRST}_1^G(\beta\alpha)$$

$$\mathrm{FIRST}_1^G(y) \subseteq \mathrm{FIRST}_1^G(\gamma\alpha)$$

и тогда заключаем, что

$$\mathrm{FIRST}_1^G(x) = \mathrm{FIRST}_1^G(y) \subseteq \mathrm{FIRST}_1^G(\beta\alpha) \cap \mathrm{FIRST}_1^G(\gamma\alpha) \neq \varnothing,$$

что противоречит первоначальному предположению о том, что условие теоремы выполнено. Достаточность и теорема доказаны.

Следствие 2.1. Теорему 2.2 можно переформулировать следующим образом: КС-грамматика G является LL(1)-грамматикой тогда и только тогда, когда для каждого множества A-правил: $A \to \alpha_1 | \alpha_2 | \ldots | \alpha_n$ — выполняются следующие условия:

1) $\mathrm{FIRST}_1^G(\alpha_i) \cap \mathrm{FIRST}_1^G(\alpha_j) = \varnothing$ *при* $i \neq j$, $1 \le i, j \le n$;

2) *если* $\alpha_i \underset{G}{\overset{*}{\Rightarrow}} \varepsilon$, *то* $\mathrm{FIRST}_1^G(\alpha_j) \cap \mathrm{FOLLOW}_1^G(A) = \varnothing$ *для всех* j: $1 \le j \le n$, $j \neq i$.

***Определение* 2.7.** Контекстно-свободная грамматика $G = (V_N, V_T, P, S)$ называется *сильной LL(k)-грамматикой*, если она удовлетворяет условию теоремы 2.2:

$$\mathrm{FIRST}_k^G(\beta\,\mathrm{FOLLOW}_k^G(A)) \cap \mathrm{FIRST}_k^G(\gamma\,\mathrm{FOLLOW}_k^G(A)) = \varnothing$$

для всех $A \in V_N$, $\beta, \gamma \in (V_N \cup V_T)^*$, таких, что существуют правила $A \to \beta$, $A \to \gamma \in P$, $\beta \neq \gamma$.

***Следствие* 2.2.** Каждая $LL(1)$-грамматика является сильной. Однако следующий пример показывает, что для $k > 1$ не всякая $LL(k)$-грамматика является сильной.

Пример 2.3. Рассмотрим cfg $G = (\{S, A\}, \{a, b\}, P, S)$, где

$P = \{S \to aAaa \mid bAba, A \to b \mid \varepsilon\}$.

Используя теорему 2.1, проверим, что грамматика G — $LL(2)$.

---

[13] Противоречие можно видеть и в том, что существуют два разных левосторонних вывода для одной и той же терминальной цепочки w.

I. Проведем тестирование относительно нетерминала S:

$\mathrm{FIRST}_2^G(aAaa\alpha) = \{aa, ab\}$,

$\mathrm{FIRST}_2^G(bAba\alpha) = \{bb\}$ независимо от α,

$\mathrm{FIRST}_2^G(aAaa\alpha) \cap \mathrm{FIRST}_2^G(bAba\alpha) = \{aa, ab\} \cap \{bb\} = \varnothing$.

II. Проведем тестирование относительно нетерминала A. При этом следует учесть, что $S \Rightarrow aAaa$ и $S \Rightarrow bAba$, так что $\alpha \in \{aa, ba\}$. Тогда при $\alpha = aa$

$\mathrm{FIRST}_2^G(baa) = \{baa\}$,

$\mathrm{FIRST}_2^G(aa) = \{aa\}$,

$\mathrm{FIRST}_2^G(baa) \cap \mathrm{FIRST}_2^G(aa) = \{baa\} \cap \{aa\} = \varnothing$;

при $\alpha = ba$

$\mathrm{FIRST}_2^G(bba) = \{bb\}$,

$\mathrm{FIRST}_2^G(ba) = \{ba\}$,

$\mathrm{FIRST}_2^G(bba) \cap \mathrm{FIRST}_2^G(ba) = \{bb\} \cap \{ba\} = \varnothing$.

Так что G — действительно $LL(2)$-грамматика.

Но поскольку, очевидно, что $\mathrm{FOLLOW}_2^G(S) = \{\varepsilon\}$, то, учитывая существование выводов $S \Rightarrow aAaa$ и $S \Rightarrow bAba$, заключаем, что $\mathrm{FOLLOW}_2^G(A) = \{aa, ba\}$.

Проверка условия сильной $LL(2)$-грамматики показывает, что для S

$\mathrm{FIRST}_2^G(aAaa\{\varepsilon\}) \cap \mathrm{FIRST}_2^G(bAba\{\varepsilon\}) = \{aa, ab\} \cap \{bb\} = \varnothing$,

а для A

$\mathrm{FIRST}_2^G(b\{aa, ba\}) \cap \mathrm{FIRST}_2^G(\varepsilon\{aa, ba\}) = \{ba, bb\} \cap \{aa, ba\} = \{ba\} \neq \varnothing$.

Таким образом, грамматика G — $LL(2)$, но не сильная $LL(2)$-грамматика.

Теорема 2.3. *Если $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика, и G — леворекурсивна, то G — не $LL(k)$-грамматика ни при каком k.*

Доказательство будем проводить в предположении приведенности грамматики G, поскольку, как отмечалось ранее, бесполезные нетерминалы не влияют на выполнение $LL(k)$-критерия.

Так как грамматика G леворекурсивна, то существует вывод вида $A \overset{+}{\underset{G}{\Rightarrow}} A\rho$, где $\rho \neq \varepsilon$, хотя и не исключается, что $\rho \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$. Напомним, что $A \overset{+}{\underset{G}{\Rightarrow}} A\rho$ означает, что существует вывод вида $A \underset{G}{\Rightarrow} \beta \overset{*}{\underset{G}{\Rightarrow}} A\rho$.

Отметим, что независимо от того, $\beta = A\rho$ или $\beta \neq A\rho$, должно существовать еще какое-то правило для A, скажем, $A \to \gamma$ ($\gamma \neq \beta$), ибо в противном случае ни-

чего, кроме $A \overset{*}{\underset{G}{\Rightarrow}} A\rho^i$, где $i \geq 0$, вывести из нетерминала A было бы невозможно, что противоречило бы приведенности грамматики G.

В любом случае вывод $A \overset{+}{\underset{G}{\Rightarrow}} A\rho$ в общем случае состоит из шагов вида

$$A \underset{G}{\Rightarrow} A_1\alpha_1 \underset{G}{\Rightarrow} A_2\alpha_2\alpha_1 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} A_{m-1}\alpha_{m-1}\ldots\alpha_1 \underset{G}{\Rightarrow} A_m\alpha_m\ldots\alpha_1 = A\rho,$$

где $A_m = A$, $\rho = \alpha_m\ldots\alpha_1$. Очевидно, что в этом выводе были использованы правила $A \to A_1\alpha_1$, $A_1 \to A_2\alpha_2, \ldots$, $A_{m-1} \to A_m\alpha_m$, где $A_m = A$. Хотя бы одно из этих правил должно иметь альтернативу, не начинающуюся ни на один из нетерминалов A_1, $A_2, \ldots$, A_m (иначе грамматика G была бы неприведенной). Пусть, например, существует $A_j \to A_{j+1}\alpha_{j+1} \mid \gamma$, где $\gamma \neq A_{j+1}\alpha_{j+1}$.

Так как грамматика G — приведенная, существует сентенциальная форма, в частности левостороннего вывода, в которой участвует нетерминал A, например $S \overset{*}{\underset{\text{lm}}{\Rightarrow}} wA\alpha$, который может быть продолжен следующим образом:

$$S \overset{*}{\underset{\text{lm}}{\Rightarrow}} wA\alpha \overset{*}{\underset{\text{lm}}{\Rightarrow}} wA\rho^i\alpha \underset{\text{lm}}{\Rightarrow} wA_1\alpha_1\rho^i\alpha \underset{\text{lm}}{\Rightarrow} \ldots \underset{\text{lm}}{\Rightarrow} wA_j\alpha_j\ldots\alpha_1\rho^i\alpha.$$

Далее этот вывод можно продолжить двумя способами:

$$1) \underset{\text{lm}}{\Rightarrow} wA_{j+1}\alpha_{j+1}\alpha_j\ldots\alpha_1\rho^i\alpha \overset{*}{\underset{\text{lm}}{\Rightarrow}} wA_m\alpha_m\ldots\alpha_1\rho^i\alpha = wA\rho^{i+1}\alpha \overset{*}{\underset{\text{lm}}{\Rightarrow}}$$

$$\overset{*}{\underset{\text{lm}}{\Rightarrow}} wA_j\alpha_j\ldots\alpha_1\rho^{i+1}\alpha \underset{\text{lm}}{\Rightarrow} w\gamma\alpha_j\ldots\alpha_1\rho^{i+1}\alpha \overset{*}{\underset{\text{lm}}{\Rightarrow}} w\underbrace{zx_j\ldots x_1r^{i+1}}_{x}t,$$

$$2) \underset{\text{lm}}{\Rightarrow} w\gamma\alpha_j\ldots\alpha_1\rho^i\alpha \overset{*}{\underset{\text{lm}}{\Rightarrow}} w\underbrace{zx_j\ldots x_1r^i}_{y}t,$$

предполагая, что ввиду приведенности грамматики существуют выводы

$$\gamma \overset{*}{\underset{\text{lm}}{\Rightarrow}} z,\ \alpha_j \overset{*}{\underset{\text{lm}}{\Rightarrow}} x_j,\ldots,\ \alpha_1 \overset{*}{\underset{\text{lm}}{\Rightarrow}} x_1,\ \rho \overset{*}{\underset{\text{lm}}{\Rightarrow}} r,\ \alpha \overset{*}{\underset{\text{lm}}{\Rightarrow}} t, \text{ где } w,z,x_j,\ldots,x_1,r,t \in V_{\text{T}}^*.$$

Эти два вывода можно сопоставить с теми, которые фигурируют в определении $LL(k)$-грамматик. Здесь $x = zx_j\ldots x_1r^{i+1}t$, а $y = wzx_j\ldots x_1r^it$.

Если $r \neq \varepsilon$, то каким бы большим ни было k, всегда путем выбора i можно добиться, чтобы $\text{FIRST}_k^G(x) = \text{FIRST}_k^G(y)$ при том, что в точке, где эти два вывода разошлись, были применены различные альтернативы для раскрытия нетерминала A_j:

$$A_j \to A_{j+1}\alpha_{j+1} \mid \gamma,\ \gamma \neq A_{j+1}\alpha_{j+1}.$$

Если $r = \varepsilon$, то имеем два разных левосторонних вывода одной и той же терминальной цепочки $wzx_j\ldots x_1t$. То и другое означает, что грамматика G — не $LL(k)$ ни при каком k.

***Следствие* 2.3.** Итак, мы имеем два достаточных признака для того, чтобы считать КС-грамматику не LL-грамматикой. Это — *неоднозначность* и *леворекурсивность*.

§ 2.3. *k*-Предсказывающие алгоритмы анализа

Для класса *LL*(*k*)-грамматик существует адекватный класс анализаторов, называемых *k-предсказывающими алгоритмами анализа*.

2.3.1. Неформальное описание

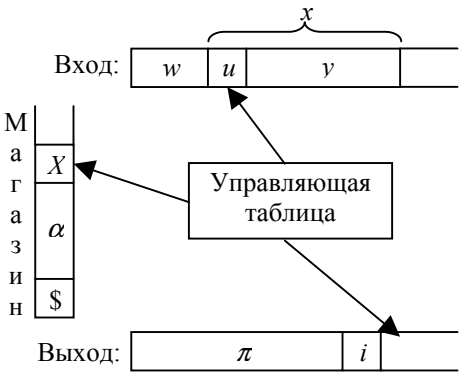

Рис. 2.1.

Это устройство (рис. 2.1) имеет разбитые на ячейки входную, выходную ленты и ленту магазина. “Дно” магазина маркируется специальным символом, например $, который постоянно находится на магазинной ленте. Конечное устройство управления руководствуется управляющей таблицей, создаваемой по грамматике, в которой производится синтаксический анализ. Она определяет действия в зависимости от *k* символов, сканируемых единовременно читающей головкой входной ленты — эта часть входной цепочки называется *аванцепочкой*, и верхнего символа магазина. Этими параметрами определяются движения следующих двух типов:

1. Запись вместо верхнего символа магазина некоторой цепочки магазинных символов и выдача на выходную ленту некоторой цепочки выходных символов. Выходная головка при этом продвигается к ближайшей свободной ячейке. В случае использования этого устройства для целей анализа на выходную ленту записываются номера правил, образующих левосторонний анализ входной цепочки, если она принимается. Этот тип движений не продвигает входную головку.

2. Сброс верхнего символа магазина и продвижение читающей головки вправо к следующей ячейке входной ленты. Такие движения происходят только тогда, когда первый символ аванцепочки — такой же, как верхний символ магазина. При этих движениях на выходную ленту ничего не пишется.

Существует возможность сигнализации о приеме входной цепочки или об ошибке в ней.

k-Предсказывающий алгоритм анализа напоминает детерминированный магазинный преобразователь, но отличается от него тем, что “видит” сразу *k* символов на входе и не имеет внутренних состояний управления. В начальный момент на входе находится анализируемая цепочка, в магазине — начальный сим-

вол магазина и маркер его "дна", на выходе пусто. Читающая головка сканирует начало входной ленты. На каждом такте работы алгоритм адресуется к своей управляющей таблице с двумя входами: верхним символом магазина и текущей аванцепочкой. Управляющий элемент диктует одно движение из двух, которое и выполняется. Этот цикл повторяется до тех пор, пока управляющий элемент не просигнализирует о приеме входной цепочки, и в этом случае на выходе образуется ее левосторонний анализ, или управляющий элемент диагностирует ошибку на входе, и тогда алгоритм останавливается, не принимая входной цепочки.

На рис. 2.1 входная цепочка представлена как wx, причем w обозначает просмотренную, а x — не просмотренную ее часть, которая начинается с аванцепочки u и заканчивается цепочкой y. Магазинная цепочка представлена в виде $X\alpha\$$, где X обозначает верхний символ магазина, α — символы магазина, располагающиеся ниже его вершины, а \$ — маркер "дна". Выходная цепочка представлена как πi, где π обозначает часть цепочки, образованную перед последним движением алгоритма, а i — последнюю произведенную запись на выходную ленту.

2.3.2. Формальное определение.

Сначала дадим несколько определений.

***Определение* 2.8.** *k-Предсказывающим алгоритмом анализа* называется формальная система $\mathfrak{A} = (\Sigma,\ \Gamma \cup \{\$\},\ \Delta,\ M,\ X_0,\ \$)$, где Σ — *входной алфавит*, $\Gamma \cup \{\$\}$ — *магазинный алфавит*, $\$ \notin \Gamma$ — *маркер* "*дна*" *магазина*, Δ — *выходной алфавит*, $X_0 \in \Gamma$ — *начальный символ магазина*, M: $(\Gamma \cup \{\$\}) \times \Sigma^{*k} \to \{(\beta, i), \text{pop}, \text{accept}, \text{error}\}$ — *управляющая таблица*, $\beta \in \Gamma^*$, $i \in \Delta$ — номер правила грамматики.

Работу k-предсказывающего алгоритма анализа проще всего описать в терминах отношения $\vdash$ на множестве конфигураций.

***Определение* 2.9.** Под *конфигурацией* k-предсказывающего алгоритма анализа будем подразумевать тройку (x, α, π), где $x \in \Sigma^*$ — непросмотренная часть входной цепочки, причем $u = \text{FIRST}_k(x) \in \Sigma^{*k}$ — аванцепочка, $\alpha \in \Gamma^*\{\$\}$ — магазинная цепочка, $\pi \in \Delta^*$ — выходная цепочка.

Начальная конфигурация есть $(w, X_0\$, \varepsilon)$, где $w \in \Sigma^*$ — вся входная цепочка.

Пусть $(x, X\alpha, \pi)$ — текущая конфигурация, где $x \in \Sigma^*$ — непросмотренная часть входной цепочки, $X \in \Gamma \cup \{\varepsilon\}$ — верхний символ магазина или пустая цепочка, $\alpha \in \Gamma^*\{\$\}$ — остальная часть магазинной цепочки.

Определим следующую конфигурацию в зависимости от значения элемента управляющей таблицы $M(X, u)$.

1. Если $M(X, u) = (\beta, i)$, то $(x, X\alpha, \pi) \vdash (x, \beta\alpha, \pi i)$.

2. Если $M(X, u) = \text{pop}$ и в этом случае всегда $X = a$, $a \in \Sigma$, $x = ax'$, $x' \in \Sigma^*$, то $(x, X\alpha, \pi) = (ax',\ a\alpha, \pi) \vdash (x', \alpha, \pi)$.

3. Если $M(X, u)$ = accept, что бывает только по достижении *конечной* или *принимающей конфигурации* $(\varepsilon, \$, \pi)$, то анализатор *останавливается*, *принимая входную цепочку*.

4. Если M(X, u) = error, то анализатор сообщает об ошибке на входе и останавливается, не принимая входной цепочки.

Как обычно, определяется степень ($\vdash^n$), транзитивное замыкание ($\vdash^+$) и рефлексивно-транзитивное замыкание ($\vdash^*$) этого отношения на конфигурациях.

Если $(w, X_0\$, \varepsilon) \vdash^* (\varepsilon, \$, \pi)$, то мы пишем $\mathfrak{A}(w) = \pi$ и называем π *выходом* $\mathfrak{A}$ для *входа* w.

Если из начальной конфигурации $(w, X_0\$, \varepsilon)$ алгоритм $\mathfrak{A}$ не достигает принимающей конфигурации, то говорят, что значение $\mathfrak{A}(w)$ *не определено*.

Трансляция, определяемая k-предсказывающим алгоритмом анализа $\mathfrak{A}$, есть $\tau(\mathfrak{A}) = \{(w, \pi) \mid \mathfrak{A}(w) = \pi\}$.

Говорят, что $\mathfrak{A}$ есть *правильный* k-предсказывающий алгоритм анализа для cfg G, если (1) $L(G) = \{w \mid \mathfrak{A}(w)$ — определен$\}$ и (2) если $\mathfrak{A}(w) = \pi$, то $S \underset{\text{lm}}{\overset{\pi}{\Rightarrow}} w$.

Если k-предсказывающий алгоритм анализа $\mathfrak{A}$, использующий управляющую таблицу M, является правильным для cfg G, то говорят, что M — *правильная управляющая таблица* для грамматики G.

Пример 2.4. Построим 1-предсказывающий алгоритм анализа для простой $LL(1)$-грамматики, приведенной в примере 2.2. Пронумеруем ее правила:

1) $S \to aBS$, 2) $S \to b$, 3) $B \to a$, 4) $B \to bSB$.

С учетом свойств этой грамматики, выявленных в примере 2.2, нетрудно построить управляющую таблицу анализатора (табл. 2.1).

Табл. 2.1

$X \in \Gamma$	Аванцепочки		
	a	b	ε
S	aBS, 1	b, 2	error
B	a, 3	bSB, 4	error
A	pop	error	error
B	error	pop	error
\$	error	error	accept

Используя эту таблицу, алгоритм $\mathfrak{A}$ анализировал бы входную цепочку *abbab* следующим образом:

$$(abbab, S\$, \varepsilon) \vdash (abbab, aBS\$, 1) \vdash (bbab, BS\$, 1) \vdash (bbab, bSBS\$, 14) \vdash$$
$$\vdash bab, SBS\$, 14) \vdash (bab, bBS\$, 142) \vdash (ab, BS\$, 142) \vdash$$
$$\vdash (ab, aS\$, 1423) \vdash (b, S\$, 1423) \vdash (b, b\$, 14232) \vdash (\varepsilon, \$, 14232).$$

Итак, $\mathfrak{A}(abbab) = 14232$. Нетрудно проверить, что 14232 — действительно левый анализ *abbab*: достаточно лишь произвести левосторонний вывод с использованием правил в указанной последовательности:

$$S \underset{\text{lm}}{\overset{1}{\Rightarrow}} aBS \underset{\text{lm}}{\overset{4}{\Rightarrow}} abSBS \underset{\text{lm}}{\overset{2}{\Rightarrow}} abbBS \underset{\text{lm}}{\overset{3}{\Rightarrow}} abbaS \underset{\text{lm}}{\overset{2}{\Rightarrow}} abbab.$$

Следовательно, $\mathfrak{A}$ — правильный 1-предсказывающий алгоритм анализа для грамматики *G*.

Пример 2.5. Построим детерминированный магазинный преобразователь, моделирующий анализатор предыдущего примера. Поскольку грамматика была простой, то нетрудно заметить, что за движением типа 1 сразу же следует pop-движение, продвигающее анализатор к следующему символу входной цепочки и стирающее верхний символ магазина, который всегда оказывается равным входному. В отличие от анализатора dpdt будет продвигаться по входной цепочке при каждом движении. Соответственно в магазин он сразу будет писать правую часть правила без первого символа. Кроме того, конец входной цепочки будет маркирован, чтобы контролировать его достижение.

Принимая во внимание сказанное, положим

$P = (\{q_0, q, \text{accept}\}, \{a, b, \$\}, \{S, B, a, b, \$\}, \{1, 2, 3, 4\}, \delta, q_0, \$, \{\text{accept}\})$,

где

1) $\delta(q_0, \varepsilon, \$) = (q, S\$, \varepsilon)$ — воспроизводит начальную конфигурацию $\mathfrak{A}$;

2) $\delta(q, a, S) = (q, BS, 1)$ — воспроизводит $M(S, a) = (aBS, 1)$;

3) $\delta(q, b, S) = (q, \varepsilon, 2)$ — воспроизводит $M(S, b) = (b, 2)$;

4) $\delta(q, a, B) = (q, \varepsilon, 3)$ — воспроизводит $M(B, a) = (a, 3)$;

5) $\delta(q, b, B) = (q, SB, 4)$ — воспроизводит $M(B, b) = (bSB, 4)$;

6) $\delta(q, \$, \$) = (\text{accept}, \varepsilon, \varepsilon)$ — сигнализирует о приеме.

Легко проверить, что $(w\$, \pi) \in \tau_e(P)$ тогда и только тогда, когда $\mathfrak{A}(w) = \pi$.

Посмотрим, как действует этот преобразователь на той же входной цепочке *abbab*:

$(q_0, abbab\$, \$, \varepsilon) \vdash (q, abbab\$, S\$, \varepsilon) \vdash (q, bbab\$, BS\$, 1) \vdash (q, bab\$, SBS\$, 14) \vdash$

$\vdash (q, ab\$, BS\$, 142) \vdash (q, b\$, S\$, 1423) \vdash (q, \$, \$, 14232) \vdash (\text{accept}, \varepsilon, \varepsilon, 14232)$.

Как видим, $(abbab\$, 14232) \in \tau_e(P)$.

§ 2.4. Построение 1-предсказывающего алгоритма анализа по *LL*(1)-грамматике

Алгоритм 2.1: построение *LL*(1)-анализатора.

Вход: $G = (V_N, V_T, P, S)$ — *LL*(1)-грамматика.

Выход: правильный $\mathfrak{A}$ — 1-предсказывающий алгоритм анализа для грамматики *G*.

Метод. Положим $\mathfrak{A} = (\Sigma, \Gamma \cup \{\$\}, \Delta, M, X_0, \$)$, где $\Sigma = V_T$, $\Delta = \{1, 2, \ldots, \#P\}$, $\Gamma = V_N \cup V_T$, $X_0 = S$.

Управляющая таблица M определяется на множестве $(\Gamma \cup \{\$\}) \times (\Sigma \cup \{\varepsilon\})$ следующим образом:

1) $M(A, a) = (\alpha, i)$, если $A \to \alpha$ является i-м правилом во множестве правил P, и $a \in \mathrm{FIRST}_1^G(\alpha)$, $a \neq \varepsilon$. Если $\varepsilon \in \mathrm{FIRST}_1^G(\alpha)$, то $M(A, b) = (\alpha, i)$ для всех $b \in \mathrm{FOLLOW}_1^G(A)$;

2) $M(a, a) = \mathrm{pop}$ для всех $a \in \Sigma$;

3) $M(\$, \varepsilon) = \mathrm{accept}$;

4) $M(X, a) = \mathrm{error}$ для всех $(X, a) \in (\Gamma \cup \{\$\}) \times (\Sigma \cup \{\varepsilon\})$, для которых значения элементов, остались не определенными по пп. 1–3.

Пример 2.6. Посредством алгоритма 2.1 построим $LL(1)$-анализатор для $LL(1)$-грамматики $G = (\{E, E', T, T', F\}, \{a, +, *, (,)\}, P, E)$, где

$P = \{(1)\ E \to TE',\ (2)\ E' \to +TE',\ (3)\ E' \to \varepsilon,\ (4)\ T \to FT',\ (5)\ T' \to *FT',$
$(6)\ T' \to \varepsilon,\ (7)\ F \to (E),\ (8)\ F \to a\}$.

Положим $\mathfrak{A} = (\{a, +, *, (,)\}, \{E, T, F, a, +, *, (,), \$\}, \{1, 2, 3, 4, 5, 6, 7, 8\}, M, E, \$)$, где M дана в виде табл. 2.2. В ней пустые клетки соответствуют значениям error.

Табл. 2.2

Маг. сим-ы	Аванцепочки					
	a	+	*	(	)	ε
E	TE', 1			TE', 1		
E'		$+TE'$, 2			ε, 3	ε, 3
T	FT', 4			FT', 4		
T'		ε, 6	$*FT'$, 5		ε, 6	ε, 6
F	a, 8			(E), 7		
a	pop					
+		pop				
*			pop			
(				pop		
)					pop	
$						accept

1-Предсказывающий алгоритм анализа, использующий эту управляющую таблицу, проанализировал бы входную цепочку $(a+a)$, совершив следующую последовательность движений:

$((a+a), E\$, \varepsilon) \vdash ((a+a), TE'\$, 1) \vdash ((a+a), FT'E'\$, 14) \vdash$
$\vdash ((a+a), (E)T'E'\$, 147) \vdash (a+a), E)T'E'\$, 147) \vdash (a+a), TE')T'E'\$, 1471) \vdash$
$\vdash (a+a), FT'E')T'E'\$, 14714) \vdash (a+a), aT'E')T'E'\$, 147148) \vdash$

$\vdash (+a),\ T'E')\ T'E'\$,\ 147148) \vdash (+a),\ E')\ T'E'\$,\ 1471486) \vdash$

$\vdash (+a),+TE')\ T'E'\$,\ 14714862) \vdash (a),\ TE')\ T'E'\$,\ 14714862) \vdash$

$\vdash (a),\ FT'E')\ T'E'\$,\ 147148624) \vdash (a),\ aT'E')\ T'E'\$,\ 1471486248) \vdash$

$\vdash (\ ),\ T'E')T'E'\$,\ 1471486248) \vdash (\ ),\ E')\ T'E'\$,\ 14714862486) \vdash$

$\vdash (\ ),\ )\ T'E'\$,\ 147148624863) \vdash (\varepsilon, T'E'\$,\ 147148624863) \vdash$

$\vdash (\varepsilon,\ E'\$,\ 1471486248636) \vdash (\varepsilon,\ \$,\ 14714862486363).$

Итак, $\mathfrak{A}((a+a)) = 14714862486363$. Нетрудно проверить, что $E \underset{\text{lm}}{\overset{\pi}{\Rightarrow}} (a+a)$, где $\pi = 14714862486363$.

Теорема 2.4. *Алгоритм* 2.1 *производит правильный* 1*-предсказывающий алгоритм анализа для любой LL*(1)*-грамматики*.

Доказательство. Пусть $G = (V_N, V_T, P, S)$ — $LL(1)$-грамматика, и $\mathfrak{A}$ — 1-предсказывающий алгоритм анализа для грамматики G, построенный согласно алгоритму 2.1. Требуется доказать, что $S \underset{\text{lm}}{\overset{\pi}{\Rightarrow}} x$ тогда и только тогда, когда $(x, S\$, \varepsilon) \overset{*}{\vdash} (\varepsilon, \$, \pi)$.

По индукции докажем вспомогательное утверждение, общий смысл которого состоит в том, что анализатор в своем магазине воспроизводит последовательность открытых частей сентенциальных форм левостороннего вывода входной цепочки, если она выводима в данной грамматике G, причем если π — последовательность номеров правил, участвующих в ее выводе, то π образуется на выходе анализатора.

I. Докажем сначала, что если $S \underset{\text{lm}}{\overset{\pi}{\Rightarrow}} x\alpha$, где $x \in \Sigma^*$— закрытая часть, а $\alpha \in (V_N \cup V_T)^*$ — открытая часть данной сентенциальной формы, то для любой цепочки $y \in \Sigma^*$, такой, что $\text{FIRST}_1(y) \in \text{FIRST}_1^G(\alpha)$, анализатор совершает переход $(xy, S\$, \varepsilon) \overset{*}{\vdash} (y, \alpha\$, \pi)$.

Индукция по $l = |\pi|$.

База. Пусть $l = 1$, т.е. $\pi = i$, где i — номер некоторого правила грамматики.

Пусть $S \underset{\text{lm}}{\overset{i}{\Rightarrow}} x\alpha$ и $y \in \Sigma^*$— такая цепочка, что $\text{FIRST}_1(y) \in \text{FIRST}_1^G(\alpha)$. На единственном шаге этого вывода применялось правило вида $S \to x\alpha$, имеющее номер i. Для всех $a \in \text{FIRST}_1^G(x\alpha\,\text{FOLLOW}_1^G(S))$ согласно алгоритму 2.1 $M(S, a) = (x\alpha, i)$

.

Посмотрим, как будет действовать анализатор, начиная с конфигурации $(xy, S\$, \varepsilon)$. Очевидно, что аванцепочка

$$u = \text{FIRST}_1(xy) \in \text{FIRST}_1^G(x\alpha) = \text{FIRST}_1^G(x\alpha\varepsilon) \subseteq \text{FIRST}_1^G(x\alpha\,\text{FOLLOW}_1^G(S))$$

и, следовательно, $M(S, u) = (x\alpha, i)$. Поэтому $(xy, S\$, \varepsilon) \vdash (xy,\ x\alpha\$,\ i) \overset{*}{\vdash} (y,\ \alpha\$, \varepsilon)$, причем последний переход реализуется посредством pop-движений. База доказана.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \le n$ ($n \ge 1$).

Индукционный переход. Докажем утверждение для $l = n + 1$. Пусть имеется левосторонний вывод длиной $n + 1$: $S \underset{\text{lm}}{\overset{\pi'}{\Rightarrow}} x'\alpha' = x'A\gamma \underset{\text{lm}}{\overset{i}{\Rightarrow}} x'\beta\gamma = x\alpha$. Здесь $\pi = \pi' i$, $\alpha' = A\gamma$, $\beta\gamma = z\alpha$, где $z \in V_T^*$, $x = x'z$, $A \in V_N$, $\beta, \gamma \in V^*$. На последнем шаге вывода применялось правило $A \to \beta$ — i-е правило из множества правил P. Согласно алгоритму 2.1

$$M(A, a) = (\beta, i) \text{ для всех } a \in \text{FIRST}_1^G(\beta\, \text{FOLLOW}_1^G(A)). \qquad (2.1)$$

Посмотрим, как будет действовать анализатор из своей начальной конфигурации $(xy, S\$, \varepsilon) = (x'zy, S\$, \varepsilon) = (x'y', S\$, \varepsilon)$, где $y' = zy$.

Применим к имеющемуся выводу $S \underset{\text{lm}}{\overset{\pi'}{\Rightarrow}} x'\alpha'$ индукционную гипотезу с цепочкой y', поскольку

$$\text{FIRST}_1(y') = \text{FIRST}_1(zy) \in \text{FIRST}_1^G(z\alpha) =$$
$$= \text{FIRST}_1^G(\beta\gamma) \subseteq \text{FIRST}_1^G(A\gamma) = \text{FIRST}_1^G(\alpha').$$

Как следствие получаем

$$(xy, S\$, \varepsilon) = (x'zy, S\$, \varepsilon) = (x'y', S\$, \varepsilon) \vdash^* (y', \alpha'\$, \pi') = (zy,\ A\gamma\$, \pi').$$

Вычислим аванцепочку от zy:

$$u = \text{FIRST}_1(zy) \in \text{FIRST}_1^G(z\alpha) =$$
$$= \text{FIRST}_1^G(\beta\gamma) \subseteq \text{FIRST}_1^G(\beta\, \text{FOLLOW}_1^G(A)),$$

а для таких аванцепочек, как показывает (2.1), $M(A, u) = (\beta, i)$. Поэтому следующее движение и завершающие pop-движения продолжают процесс таким образом:

$$(zy, A\gamma\$, \pi') \vdash (zy, \beta\gamma\$, \pi' i) = (zy, z\alpha\$, \pi' i) \vdash^* (y, \alpha\$, \pi).$$

Что и требовалось.

II. Докажем теперь, что если анализатор совершает переход вида $(xy, S\$, \varepsilon) \vdash^* (y, \alpha\$, \pi)$ для любой цепочки $y \in \Sigma^*$, такой, что $\text{FIRST}_1(y) \in \text{FIRST}_1^G(\alpha)$, то $S \underset{\text{lm}}{\overset{\pi}{\Rightarrow}} x\alpha$, где x — закрытая часть, а α — открытая часть данной сентенциальной формы.

Индукция по $l = |\pi|$.

База. Пусть $l = 1$, т.е. $\pi = i$.

Имеем $(xy, S\$, \varepsilon) \vdash^* (y, \alpha\$, i)$. Анализатор пишет на выходную ленту номер правила только при движении типа 1, а оно совершается только при наличии нетерминала на вершине магазина. Следовательно, это — первое движение, и только оно пишет номер i на выход. Поэтому фактически имеем

$$(xy, S\$, \varepsilon) \vdash (xy, \beta\$, \varepsilon) \vdash^* (y, \alpha\$, i),$$

причем все остальные движения — это pop-движения. Очевидно, что только при $\beta = x\alpha$ достижима завершающая конфигурация. Первое движение обеспечивается элементом таблицы $M(S, a) = (\beta, i)$, где $u = \mathrm{FIRST}_1(xy)$, а это подразумевает существование правила $S \to \beta = x\alpha$ под номером i. Чтобы первое движение могло произойти, необходимо, чтобы $u \in \mathrm{FIRST}_1^G(x\alpha\ \mathrm{FOLLOW}_1^G(S))$. И это действительно так, поскольку

$$u = \mathrm{FIRST}_1(xy) \in \mathrm{FIRST}_1^G(x\alpha\varepsilon) \subseteq \mathrm{FIRST}_1^G(x\alpha\,\mathrm{FOLLOW}_1^G(S)).$$

А тогда $S \underset{\text{lm}}{\overset{i}{\Longrightarrow}} x\alpha$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \le n$ ($n \ge 1$).

Индукционный переход. Докажем утверждение для $l = n + 1$.

Пусть

$$(xy, S\$, \varepsilon) = (x'y', S\$, \varepsilon) \vdash^* (y', \alpha'\$, \pi') = (y', A\gamma\$, \pi') \vdash (y', \beta\gamma\$, \pi'i) \vdash^* (y, \alpha\$, \pi),$$

где $\pi = \pi' i$, $|\pi'| = n$, $xy = x'y'$, $\alpha' = A\gamma$. Завершающие pop-движения показывают, что $y' = zy$, $\beta\gamma = z\alpha$; т.к. $xy = x'y' = x'zy$, то $x = x'z$.

Последнее движение типа 1 было выполнено благодаря элементу $M(A, u') = (\beta, i)$ для $u' \in \mathrm{FIRST}_1(y')$, что предполагает существование правила $A \to \beta$ под номером i, а также выполнение условия $u' \in \mathrm{FIRST}_1^G(\beta\,\mathrm{FOLLOW}_1^G(A))$. Оно действительно выполняется, поскольку

$$u' \in \mathrm{FIRST}_1(y') = \mathrm{FIRST}_1(zy) \in \mathrm{FIRST}_1^G(z\alpha) = \mathrm{FIRST}_1^G(\beta\gamma) \subseteq \mathrm{FIRST}_1^G(\beta\,\mathrm{FOLLOW}_1^G(A)).$$

Одновременно можно воспользоваться индукционной гипотезой в применении к $(x'y', S\$, \varepsilon) \vdash^* (y', \alpha'\$, \pi')$, поскольку

$$\mathrm{FIRST}_1(y') = \mathrm{FIRST}_1(zy) \in \mathrm{FIRST}_1^G(z\alpha) = \mathrm{FIRST}_1^G(\beta\gamma) \subseteq \mathrm{FIRST}_1^G(A\gamma) = \mathrm{FIRST}_1^G(\alpha'),$$

и получить как следствие $S \underset{\text{lm}}{\overset{\pi'}{\Longrightarrow}} x'\alpha' = x'A\gamma \underset{\text{lm}}{\overset{i}{\Longrightarrow}} x'\beta\gamma = x'z\alpha = x\alpha$. Что и требовалось.

Из утверждений I и II при $y = \alpha = \varepsilon$ заключаем, что $S \underset{\text{lm}}{\overset{\pi}{\Longrightarrow}} x$ тогда и только тогда, когда $(x, S\$, \varepsilon) \vdash^* (\varepsilon, \$, \pi)$. А это и означает, что 1-предсказывающий алгоритм анализа, построенный согласно алгоритму 2.1, является правильным для $LL(1)$-грамматики G. Теорема доказана.

Замечание 2.1. Алгоритм 2.1 пригоден для построения анализаторов для $LL(k)$-грамматик и при $k > 1$, если только они сильные. При его применении к сильным $LL(k)$-грамматикам всюду, где используется параметр 1, следует использовать значение k.

Прежде чем перейти к обсуждению *LL*-анализа при $k > 1$, введем в рассмотрение еще одну полезную операцию над языками.

***Определение* 2.10.** Пусть Σ — некоторый алфавит и L_1, L_2 — подмножества Σ^*. Положим

$$L_1 \oplus_k L_2 = \left\{ w \in \Sigma^{*k} \;\middle|\; x \in L_1, y \in L_2, w = \begin{cases} xy, \text{ если } |xy| \le k, \\ \mathrm{FIRST}_k(xy) \text{ в противном случае} \end{cases} \right\}.$$

Пример 2.7. Пусть $L_1 = \{\varepsilon, abb\}$ и $L_2 = \{b, bab\}$. Тогда $L_1 \oplus_2 L_2 = \{b, ba, ab\}$. Операция $\oplus_k$ подобна инфиксной операции FIRST_k.

Лемма 2.1. *Для любой контекстно-свободной грамматики* $G = (V_N, V_T, P, S)$ *и любых цепочек* $\alpha, \beta \in V^*$ *имеет место тождество*

$$\mathrm{FIRST}_k^G(\alpha\beta) = \mathrm{FIRST}_k^G(\alpha) \oplus_k \mathrm{FIRST}_k^G(\beta).$$

Доказательство.

I. Пусть $w \in \mathrm{FIRST}_k^G(\alpha\beta)$. Докажем, что тогда $w \in \mathrm{FIRST}_k^G(\alpha) \oplus_k \mathrm{FIRST}_k^G(\beta)$.

Обозначим $L_1 = \mathrm{FIRST}_k^G(\alpha)$, $L_2 = \mathrm{FIRST}_k^G(\beta)$. Из того, что $w \in \mathrm{FIRST}_k^G(\alpha\beta)$, следует, что $w = \mathrm{FIRST}_k(uv)$, если $\alpha \overset{*}{\Rightarrow} u$, $\beta \overset{*}{\Rightarrow} v$. Последние два вывода означают, что $x = \mathrm{FIRST}_k(u) \in \mathrm{FIRST}_k^G(\alpha) = L_1$.

Аналогично $y = \mathrm{FIRST}_k(v) \in \mathrm{FIRST}_k^G(\beta) = L_2$. Учитывая определение операции $\oplus_k$, заключаем, что $\mathrm{FIRST}_k(xy) \in L_1 \oplus_k L_2$.

Остается убедиться в том, что $\mathrm{FIRST}_k(xy) = \mathrm{FIRST}_k(uv) = w$. Так как $x = \mathrm{FIRST}_k(u)$, а $y = \mathrm{FIRST}_k(v)$, то $u = xu'$ и $v = yv'$ при некоторых $u', v' \in V_T^*$, причем если $|x| < k$, то $u' = \varepsilon$, и если $|y| < k$, то $v' = \varepsilon$. Итак, имеем $uv = xu'yv'$.

Если $|x| = k$, то $w = \mathrm{FIRST}_k(uv) = x = \mathrm{FIRST}_k(xy)$. Если $|x| < k$, то $u' = \varepsilon$ и $w = \mathrm{FIRST}_k(uv) = \mathrm{FIRST}_k(xyv')$, причем, если $|y| < k$, то $v' = \varepsilon$ и тогда $w = \mathrm{FIRST}_k(uv) = \mathrm{FIRST}_k(xy)$, если же $|y| = k$, то $\mathrm{FIRST}_k(uv) = \mathrm{FIRST}_k(xyv') = \mathrm{FIRST}_k(xy)$. Итак,

$$w = \mathrm{FIRST}_k(uv) = \mathrm{FIRST}_k(xy) \in L_1 \oplus_k L_2 = \mathrm{FIRST}_k^G(\alpha) \oplus_k \mathrm{FIRST}_k^G(\beta).$$

II. Пусть $w \in \mathrm{FIRST}_k^G(\alpha) \oplus_k \mathrm{FIRST}_k^G(\beta)$. Докажем, что тогда $w \in \mathrm{FIRST}_k^G(\alpha\beta)$.

Согласно определению операции $\oplus_k$ существуют цепочки $x \in \mathrm{FIRST}_k^G(\alpha)$, $y \in \mathrm{FIRST}_k^G(\beta)$ и $w = \mathrm{FIRST}_k(xy)$. Кроме того, согласно определению FIRST_k^G имеем $\alpha \overset{*}{\Rightarrow} xu$, $\beta \overset{*}{\Rightarrow} yv$ при некоторых $u, v \in V_T^*$ и $\mathrm{FIRST}_k(xuyv) \in \mathrm{FIRST}_k^G(\alpha\beta)$. Остается убедиться в том, что $\mathrm{FIRST}_k(xuyv) = \mathrm{FIRST}_k(xy)$.

Если $|x| = k$, то $\mathrm{FIRST}_k(xuyv) = \mathrm{FIRST}_k(xy) = x$. Если $|x| < k$, то $u = \varepsilon$ и $\mathrm{FIRST}_k(xuyv) = \mathrm{FIRST}_k(xyv)$, причем, если $|y| < k$, то $v = \varepsilon$ и $\mathrm{FIRST}_k(xuyv) = \mathrm{FIRST}_k(xy)$, в противном случае от v ничего не зависит и $\mathrm{FIRST}_k(xuyv) = \mathrm{FIRST}_k(xy)$. Следовательно, $w = \mathrm{FIRST}_k(xy) = \mathrm{FIRST}_k(xuyv) \in \mathrm{FIRST}_k(\alpha\beta)$.

Из рассуждений I и II следует утверждение леммы. Что и требовалось.

§ 2.5. Анализ в *LL*(*k*)-грамматиках

В общем случае анализа в *LL*(*k*)-грамматиках по нетерминалу и аванцепочке невозможно однозначно определить правило для построения очередной сентенциальной формы. Рассмотрим, например, *LL*(2)-грамматику из примера 2.3: $S \to aAaa \mid bAba;\ \ A \to b \mid \varepsilon$.

Согласно алгоритму 2.1 для замены нетерминала A в выводе $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha \underset{lm}{\overset{*}{\Rightarrow}} wx$ следует применять правило $A \to \beta$, если аванцепочка $u = \mathrm{FIRST}_2^G(x)$ принадлежит множеству $\mathrm{FIRST}_2^G(\beta\,\mathrm{FOLLOW}_2^G(A))$. Очевидно, что в нашем примере $\mathrm{FOLLOW}_2^G(A) = \{aa, ba\}$. Получается так, что следует применять правило $A \to b$, если аванцепочка из $\mathrm{FIRST}_2^G(b\,\mathrm{FOLLOW}_2^G(A)) = \mathrm{FIRST}_2^G(b\{aa, ba\}) = \{ba, bb\}$, или правило $A \to \varepsilon$, если аванцепочка из множества $\mathrm{FIRST}_2^G(\varepsilon\,\mathrm{FOLLOW}_2^G(A)) = \{aa, ba\}$. Соображения о том, как определять, какое правило из этих двух использовать для замены нетерминала A, если аванцепочка равна ba, уже были рассмотрены в упомянутом примере. Эту неопределенность можно разрешить, если для определения правила, по которому следует раскрывать нетерминал, помимо нетерминала и аванцепочки использовать еще один параметр: $T_{A,L}$ — так называемую *LL*(*k*)-таблицу[14], ассоциированную с двумя индексами, первый из которых — нетерминал A, второй — множество L, вычисляемое с учетом следующих рассуждений.

Пусть имеется вывод $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha$ в некоторой *LL*(*k*)-грамматике. Вспоминая теорему 2.1, нетрудно сообразить, что критерием выбора правила $A \to \beta$ для продолжения этого вывода может служить факт принадлежности текущей аванцепочки множеству

$$\mathrm{FIRST}_k^G(\beta\alpha) = \mathrm{FIRST}_k^G(\beta) \oplus_k \mathrm{FIRST}_k^G(\alpha) = \mathrm{FIRST}_k^G(\beta) \oplus_k L,$$

где $L = \mathrm{FIRST}_k^G(\alpha)$. Это множество L и есть тот второй индекс таблицы $T_{A,L}$, о котором шла речь. По аванцепочке таблица $T_{A,L}$ выдает правило, которое следует использовать для замены нетерминала A в текущей левосентенциальной форме.

[14] Не путать с управляющей таблицей *k*-предсказывающего алгоритма анализа.

Ясно, что для любой cfg число таких $LL(k)$-таблиц конечно и все таблицы, необходимые для анализа в данной грамматике, могут быть построены заблаговременно. Они используются k-предсказывающим алгоритмом анализа в магазинных цепочках взамен нетерминалов (см. алгоритм 2.2 далее).

Теперь дадим точное определение $LL(k)$-таблиц и опишем способ построения множества $LL(k)$-таблиц, необходимых для анализа в данной $LL(k)$-грамматике.

***Определение* 2.11.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. Для каждого $A \in V_N$ и $L \subseteq V_T^{*k}$ определим $T_{A,L}$ — LL(k)-таблицу, ассоциированную с A и L как функцию, которая по данной аванцепочке $u \in V_T^{*k}$ выдает либо значение error, либо A-правило[15] и конечный список подмножеств V_T^{*k}. Именно:

1) $T_{A,L}(u) =$ error, если не существует ни одного правила вида $A \to \alpha \in P$, такого, что $u \in \mathrm{FIRST}_k^G(\alpha) \oplus_k L$;

2) $T_{A,L}(u) = (A \to \alpha, \langle Y_1, Y_2, \ldots, Y_m \rangle)$, если $A \to \alpha$ — единственное правило из P, такое, что $u \in \mathrm{FIRST}_k^G(\alpha) \oplus_k L$; при этом, если $\alpha = x_0 B_1 x_1 B_2 \ldots B_m x_m$, $B_i \in V_N$, $x_j \in V_T^*$, то $Y_i = \mathrm{FIRST}_k^G(x_i B_{i+1} x_{i+1} \ldots B_m x_m) \oplus_k L$, $i = 1,2,\ldots,m$; $j = 0,1,\ldots,m$; $m \geq 0$;

3) $T_{A,L}(u) =$ undefined, если существует несколько A-правил: $A \to \alpha_1 | \alpha_2 | \ldots | \alpha_n$, таких, что $u \in \mathrm{FIRST}_k^G(\alpha_i) \oplus_k L$ для $1 \leq i \leq n$, $n \geq 2$.

Множество терминальных цепочек Y_i называется *множеством локальных правых контекстов для* B_i. В частности, если $m = 0$, то $T_{A,L}(u) = (A \to \alpha, \varnothing)$. Случай 3 для $LL(k)$-грамматик не актуален.

Пусть мы имеем левосторонний вывод в $LL(k)$-грамматике: $S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\gamma \underset{\text{lm}}{\overset{*}{\Rightarrow}} wx$. Как было сказано в начале этого параграфа, именно таблица $T_{A,L}$, где $L = \mathrm{FIRST}_k^G(\gamma)$, сообщит, что следующую за $wA\gamma$ сентенциальную форму следует получать при помощи правила $A \to \alpha$, если для $u = \mathrm{FIRST}_k(x)$ окажется, что $T_{A,L}(u) = (A \to \alpha, \langle Y_1, Y_2, \ldots, Y_m \rangle)$. Если $\alpha = x_0 B_1 x_1 B_2 \ldots B_m x_m$, то следующая за $wA\gamma$ сентенциальная форма будет $w\alpha\gamma = wx_0 B_1 x_1 B_2 \ldots B_m x_m \gamma$, и в свое время раскрытием B_1 будет руководить $LL(k)$-таблица T_{B_1,Y_1}, а раскрытием B_m — $LL(k)$-таблица T_{B_m,Y_m}.

Пример 2.8. Рассмотрим уже не раз обсуждавшуюся $LL(2)$-грамматику с правилами $S \to aAaa \mid bAba$; $A \to b \mid \varepsilon$ и построим все $LL(2)$-таблицы, необходимые для анализа в этой грамматике.

Начнем с таблицы $T_{S,\{\varepsilon\}}$: именно она диктует выбор первого правила для раскрытия нетерминала S. Используя правило $S \to aAaa$, вычисляем $\mathrm{FIRST}_2^G(aAaa) \oplus_2 \{\varepsilon\} = \{ab, aa\}$. Используя правило $S \to bAba$, вычисляем $\mathrm{FIRST}_2^G(bAba) \oplus_2 \{\varepsilon\} = \{bb\}$. Соответственно определяем $LL(2)$-таблицу: $T_0 = T_{S,\{\varepsilon\}}$ (табл. 2.3).

[15] A-Правило — правило cfg с нетерминалом A в левой части.

Табл. 2.3

$LL(2)$-табл.	u	$T(u)$	
		Правило	Мн-во лок. прав. конт-в
$T_0 = T_{S,\{\varepsilon\}}$	aa ab bb	$S \to aAaa$ $S \to aAaa$ $S \to bAba$	$\langle\{\, aa \,\}\rangle$ $\langle\{\, aa \,\}\rangle$ $\langle\{\, ba \,\}\rangle$
$T_1 = T_{A,\{aa\}}$	ba aa	$A \to b$ $A \to \varepsilon$	— —
$T_2 = T_{A,\{ba\}}$	bb ba	$A \to b$ $A \to \varepsilon$	— —

Глядя на T_0, легко определить, что потребуются еще таблицы $T_1 = T_{A,\{aa\}}$ и $T_2 = T_{A,\{ba\}}$. Таблица T_1 получается с учетом того, что для правила $A \to b$ получаем $\mathrm{FIRST}_2^G(b) \oplus_2 \{aa\} = \{ba\}$, а для правила $A \to \varepsilon$ — $\mathrm{FIRST}_2^G(\varepsilon) \oplus_2 \{aa\} = \{aa\}$. Таблица T_2 получается с учетом того, что для правила $A \to b$ имеем $\mathrm{FIRST}_2^G(b) \oplus_2 \{ba\} = \{bb\}$, а для правила $A \to \varepsilon$ — $\mathrm{FIRST}_2^G(\varepsilon) \oplus_2 \{ba\} = \{ba\}$. В правых частях правил для нетерминала A нет ни одного нетерминала. Поэтому в двух последних таблицах множества локальных правых контекстов пусты.

При построении этих $LL(2)$-таблиц мы придерживались простой дисциплины: начинали с построения начальной таблицы $T_{S,\{\varepsilon\}}$, а затем в каждой из уже построенных $LL(2)$-таблиц использовали значения элементов, чтобы определить пары индексов других необходимых таблиц — каждое вхождение нетерминала в правило сочеталось с соответствующим локальным множеством. Этот порядок построения $LL(k)$-таблиц фиксируется в следующем описании алгоритма:

Алгоритм 2.2: построение множества $LL(k)$-таблиц, необходимых для анализа в данной $LL(k)$-грамматике.

Вход: $G = (V_N, V_T, P, S)$ — $LL(k)$-грамматика.

Выход: $\mathcal{T}$ — множество $LL(k)$-таблиц, необходимых для анализа в грамматике G.

Метод.

1. Построить $T_0 = T_{S,\{\varepsilon\}}$ и $\mathcal{T} = \{T_0\}$.

2. Если $T_{A,L} \in \mathcal{T}$ и для некоторой цепочки $u \in V_T^{*k}$ и $T_{A,L}(u) = (A \to x_0 B_1 x_1 B_2 \ldots B_m x_m, \langle Y_1, Y_2, \ldots, Y_m \rangle)$, то к множеству таблиц $\mathcal{T}$ добавить те таблицы из множества $\{T_{B_i,Y_i} \mid i = 1, 2, \ldots, m\}$, которых нет в $\mathcal{T}$.

3. Повторять шаг 2 до тех пор, пока ни одну новую $LL(k)$-таблицу не удастся добавить к $\mathcal{T}$.

Такой момент обязательно настанет, так как для любой данной cfg G существует только конечное число таких таблиц (число нетерминалов — конечно, число подмножеств $L \subseteq V_{\mathrm{T}}^{*k}$ тоже конечно). Фактически же для анализа требуются не все возможные $LL(k)$-таблицы, которые можно построить для грамматики G, а только те, которые определяются описанным алгоритмом.

Алгоритм 2.3: построение k-предсказывающего алгоритма анализа.

Вход: $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, P, S)$ — $LL(k)$-грамматика.

Выход: $\mathfrak{A}$ — правильный k-предсказывающий алгоритм анализа для G.

Метод.

1. Построим $\mathscr{T}$— множество необходимых $LL(k)$-таблиц для грамматики G.

2. Положим $\mathfrak{A} = (\Sigma, \Gamma \cup \{\$\}, \Delta, M, T_0, \$)$, где $\Sigma = V_{\mathrm{T}}$, $\Delta = \{1, 2, \ldots, \#P\}$, $\Gamma = \mathscr{T} \cup V_{\mathrm{T}}$, где $T_0 = T_{S,\{\varepsilon\}}$.

3. Управляющую таблицу M определим на множестве $(\Gamma \cup \{\$\}) \times \Sigma^{*k}$ следующим образом:

3.1. $M(T_{A,L}, u) = (x_0 T_{B_1,Y_1} x_1 T_{B_2,Y_2} \ldots T_{B_m,Y_m} x_m, i)$, если $T_{A,L}(u) = (A \to x_0 B_1 x_1 B_2 \ldots B_m x_m, \langle Y_1, Y_2, \ldots, Y_m \rangle)$, и $A \to x_0 B_1 x_1 B_2 \ldots B_m x_m$ является i-м правилом в P.

3.2. $M(a, av) =$ pop для всех $a \in \Sigma$, $v \in \Sigma^{*k}$.

3.3. $M(\$, \varepsilon) =$ accept.

3.4. $M(X, u) =$ error для всех $(X, u) \in (\Gamma \cup \{\$\}) \times \Sigma^{*k}$, для которых значения элементов, остались не определенными по пп. 1–3.

Пример 2.9. Рассмотрим еще раз $LL(2)$-грамматику, обсуждавшуюся в примере 2.3, с правилами

1) $S \to aAaa$, 2) $S \to bAba$, 3) $A \to b$, 4) $A \to \varepsilon$.

Используя уже построенные для нее $LL(2)$-таблицы легко собрать управляющую таблицу для этой грамматики — см. табл. 2.4.

Табл. 2.4

Маг. сим-ы	Аванцепочки						
	aa	*ab*	*ba*	*bb*	*a*	*b*	ε
T_0	aT_1aa, 1	aT_1aa, 1		bT_2ba, 2			
T_1	ε, 4		*b*, 3				
T_2			ε, 4	*b*, 3			
a	pop	pop			pop		
b			pop	pop		pop	
\$							accept

Например, на входной цепочке *bba* этот 2-предсказывающий алгоритм анализа проходит следующие конфигурации:

$$(bba, T_0\$, \varepsilon) \vdash (bba, bT_2ba\$, 2) \vdash (ba, T_2ba\$, 1) \vdash (ba, ba\$, 24) \vdash$$
$$\vdash (a, a\$, 24) \vdash (\varepsilon, \$, 24).$$

В то же время, посредством правил 1 и 2, получаем $S \underset{\text{lm}}{\overset{2}{\Rightarrow}} bAba \underset{\text{lm}}{\overset{4}{\Rightarrow}} bba$.

Теорема 2.5. *Алгоритм* 2.3 *производит правильный k-предсказывающий алгоритм анализа для любой LL(k)-грамматики.*

Доказательство аналогично доказательству предыдущей теоремы. Пусть $G = (V_N, V_T, P, S)$ — $LL(k)$-грамматика и $\mathfrak{A}$ — k-предсказывающий алгоритм анализа для грамматики G, построенный посредством алгоритма 2.2.

Требуется доказать, что $S \underset{lm}{\overset{\pi}{\Rightarrow}} x$ тогда и только тогда, когда $(x, T_0\$, \varepsilon) \overset{*}{\vdash} (\varepsilon, \$, \pi)$.

По индукции докажем вспомогательное утверждение, общий смысл которого состоит в том, что анализатор в своем магазине воспроизводит последовательность образов открытых частей сентенциальных форм левостороннего вывода входной цепочки, если она выводима в данной грамматике G, причем если π — последовательность номеров правил, участвующих в ее выводе, то π образуется на выходе анализатора. Связь между открытыми частями сентенциальных форм и их образами в магазине $LL(k)$-анализатора описывается следующим гомоморфизмом:

$$h(x) = \begin{cases} a, \text{ если } X = a, \ a \in V_T, \\ A, \text{ если } X = T_{A,L} \in \mathcal{T}. \end{cases}$$

Область определения h легко расширить до Γ^*, и мы будем в дальнейшем предполагать, что это сделано.

I. Докажем сначала, что если $S \underset{lm}{\overset{\pi}{\Rightarrow}} x\alpha$, где x — закрытая, а α — открытая часть данной сентенциальной формы, то для любой цепочки $y \in \Sigma^*$, такой, что $\mathrm{FIRST}_k(y) \in \mathrm{FIRST}_k^G(\alpha)$, анализатор совершает переход $(xy, T_0\$, \varepsilon) \overset{*}{\vdash} (y, \tilde{\alpha}\$, \pi)$, где $h(\tilde{\alpha}) = \alpha$. Попросту говоря, если в $\tilde{\alpha}$ заменить $LL(k)$-таблицы на нетерминалы, с которыми они ассоциированы, то получится α.

Индукция по $l = |\pi|$.

База. Пусть $l = 1$, т.е. $\pi = i$, где i — номер некоторого правила грамматики.

Пусть $S \underset{lm}{\overset{i}{\Rightarrow}} x\alpha$ и $y \in \Sigma^*$, такая, что $\mathrm{FIRST}_k(y) \in \mathrm{FIRST}_k^G(\alpha)$. На единственном шаге этого вывода применяется правило вида $S \to x\alpha$, имеющее номер i. Согласно алгоритму 2.3 $M(S, u) = (x\tilde{\alpha}, i)$ для всех $u \in \mathrm{FIRST}_k^G(x\alpha) \oplus_k \{\varepsilon\}$. Посмотрим, как будет действовать $LL(k)$-анализатор, начиная с конфигурации $(xy, T_0\$, \varepsilon)$.

Очевидно, что аванцепочка $u = \mathrm{FIRST}_k(xy) \in \mathrm{FIRST}_k^G(x\alpha) = \mathrm{FIRST}_k^G(x\alpha) \oplus_k \{\varepsilon\}$ и, следовательно, $M(T_0, u) = (x\tilde{\alpha}, i)$. Поэтому $(xy, T_0\$, \varepsilon) \vdash (xy, x\tilde{\alpha}\$, i) \overset{*}{\vdash} (y, \tilde{\alpha}\$, \varepsilon)$, причем последний переход происходит посредством pop-движений. База доказана.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \le n$ $(n \ge 1)$.

Индукционный переход. Докажем утверждение для $l = n + 1$. Пусть имеется левосторонний вывод длиной $n + 1$: $S \underset{lm}{\overset{\pi'}{\Rightarrow}} x'\alpha' = x'A\gamma \underset{lm}{\overset{i}{\Rightarrow}} x'\beta\gamma = x\alpha$. Здесь

$\pi = \pi' i$, $\alpha' = A\gamma$, $\beta\gamma = z\alpha$, где $z \in V_T^*$, $x = x'z$, $A \in V_N$, $\beta, \gamma \in V^*$. На последнем шаге вывода применялось i-е правило $A \to \beta$ — из множества P.

Согласно алгоритму 2.3

$$M(T_{A,L}, u) = (\tilde{\beta}, i) \text{ для всех } u \in \mathrm{FIRST}_k^G(\beta) \oplus_k L, \text{ где } L = \mathrm{FIRST}_k^G(\gamma). \qquad (2.2)$$

Посмотрим, как будет действовать анализатор из своей начальной конфигурации $(xy, T_0\$, \varepsilon) = (x'zy, T_0\$, \varepsilon) = (x'y', T_0\$, \varepsilon)$, где $y' = zy$. Применим к имеющемуся выводу $S \underset{lm}{\overset{\pi'}{\Rightarrow}} x'\alpha'$ индукционную гипотезу с цепочкой $y' = zy$, поскольку

$$\mathrm{FIRST}_k(y') = \mathrm{FIRST}_k(zy) \in \mathrm{FIRST}_k^G(z\alpha) = \mathrm{FIRST}_k^G(\beta\gamma) \subseteq$$

$$\subseteq \mathrm{FIRST}_k^G(A\gamma) = \mathrm{FIRST}_k^G(\alpha').$$

Как следствие получаем

$$(xy, T_0\$, \varepsilon) = (x'zy, T_0\$, \varepsilon) = (x'y', T_0\$, \varepsilon) \vdash^* (y', \tilde{\alpha}'\$, \pi') = (zy, T_{A,L}\tilde{\gamma}\$, \pi'). \qquad (2.3)$$

Вычислим аванцепочку

$$u = \mathrm{FIRST}_k(zy) \in \mathrm{FIRST}_k^G(z\alpha) = \mathrm{FIRST}_k^G(\beta\gamma) =$$

$$= \mathrm{FIRST}_k^G(\beta) \oplus_k \mathrm{FIRST}_k^G(\gamma) = \mathrm{FIRST}_k^G(\beta) \oplus_k L,$$

а для таких аванцепочек, как показывает (2.2), $M(T_{A,L}, u) = (\tilde{\beta}, i)$. Поэтому следующее движение и завершающие pop-движения продолжают процесс (2.3) следующим образом:

$$(zy,\ T_{A,L}\tilde{\gamma}\$, \pi') \vdash (zy, \tilde{\beta}\tilde{\gamma}\$, \pi' i) = (zy, z\tilde{\alpha}\$, \pi' i) \vdash^* (y, \tilde{\alpha}\$, \pi).$$

Что и требовалось.

II. Докажем теперь, что если $LL(k)$-анализатор совершает переход вида $(xy, T_0\$, \varepsilon) \vdash^* (y, \tilde{\alpha}\$, \pi)$ для любой цепочки $y \in \Sigma^*$, такой, что $\mathrm{FIRST}_k(y) \in \mathrm{FIRST}_k^G(\alpha)$, где $\alpha = h(\tilde{\alpha})$, то $S \underset{lm}{\overset{\pi}{\Rightarrow}} x\alpha$, где x — закрытая, а α — открытая часть данной сентенциальной формы.

Индукция по $l = |\pi|$.

База. Пусть $l = 1$, т.е. $\pi = i$.

Тогда $(xy,\ T_0\$,\ \varepsilon) \vdash^* (y,\ \tilde{\alpha}\$,\ i)$. Анализатор пишет на выходную ленту номер правила только при движении типа 1, а оно совершается только при наличии $LL(k)$-таблицы на вершине магазина. Следовательно, это — первое движение, и только оно пишет номер i на выход. Поэтому фактически имеем

$$(xy,\ T_0\$,\ \varepsilon) \vdash (xy,\ \tilde{\beta}\$,\ \varepsilon) \vdash^* (y,\ \tilde{\alpha}\$,\ i),$$

причем все остальные движения — это pop-движения. Очевидно, что только при $\tilde{\beta} = x\tilde{\alpha}$ достижима завершающая конфигурация посредством одних только pop-движений. Первое движение обеспечивалось элементом таблицы $M(T_0, u) = (\tilde{\beta}, i)$, где $u = \mathrm{FIRST}_k(xy)$, а это подразумевает существование правила $S \to \beta$, в котором $\beta = x\alpha$, под номером i. Тогда $S \underset{lm}{\overset{i}{\Rightarrow}} x\alpha$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $l \le n$ $(n \ge 1)$.

Индукционный переход. Докажем утверждение для $l = n + 1$.

Пусть

$$(xy, T_0\$, \varepsilon) = (x'y', T_0\$, \varepsilon) \vdash^* (y', \tilde{\alpha}'\$, \pi') =$$
$$= (y', T_{A,L}\tilde{\gamma}\$, \pi') \vdash (y', \tilde{\beta}\tilde{\gamma}\$, \pi' i) \vdash^* (y, \tilde{\alpha}\$, \pi),$$

где $\pi = \pi' i$, $|\pi'| = n$. Завершающие pop-движения предполагают, что $y' = zy$, $\beta\gamma = z\alpha$. Кроме того, мы имеем $xy = x'y' = x'zy$, откуда $x = x'z$. Так как $\tilde{\alpha}' = T_{A,L}\tilde{\gamma}$, то последнее движение типа 1 было выполнено благодаря элементу $M(T_{A,L}, u) = (\tilde{\beta}, i)$ для $u \in \mathrm{FIRST}_k(y')$, что предполагает существование правила $A \to \beta$ под номером i.

Воспользовавшись индукционной гипотезой в применении к переходу $(x'y', T_0\$, \varepsilon) \vdash^* (y', \alpha'\$, \pi')$, поскольку

$$\mathrm{FIRST}_k(y') = \mathrm{FIRST}_k(zy) \in \mathrm{FIRST}_k^G(z\alpha) = \mathrm{FIRST}_k^G(\beta\gamma) \subseteq$$
$$\subseteq \mathrm{FIRST}_k^G(A\gamma) = \mathrm{FIRST}_k^G(\alpha'),$$

получаем как следствие $S \underset{\text{lm}}{\overset{\pi'}{\Rightarrow}} x'\alpha' = x'A\gamma \underset{\text{lm}}{\overset{i}{\Rightarrow}} x'\beta\gamma = x'z\alpha = x\alpha$. Здесь α, β, γ — гомоморфные образы $\tilde{\alpha}$, $\tilde{\beta}$, $\tilde{\gamma}$ соответственно. Что и требовалось.

Из рассуждений I и II при $y = \alpha = \varepsilon$ заключаем, что $S \underset{\text{lm}}{\overset{\pi}{\Rightarrow}} x$ тогда и только тогда, когда $(x, T_0\$, \varepsilon) \vdash^* (\varepsilon, \$, \pi)$. А это и означает, что k-предсказывающий алгоритм анализа, построенный согласно алгоритму 2.3, является правильным для $LL(k)$-грамматики G. Что и требовалось доказать.

Теорема 2.6. *Число шагов, выполняемых k-предсказывающим алгоритмом анализа, линейно зависит от длины анализируемой цепочки.*

Доказательство. Поскольку k-предсказывающий алгоритм анализа применим только к $LL(k)$-грамматикам, а они не могут быть леворекурсивными (теорема 2.3), то максимальное число шагов в любом левостороннем выводе вида $A \underset{\text{lm}}{\overset{+}{\Rightarrow}} B\alpha$ меньше, чем некоторая константа c, зависящая от грамматики. Во всяком случае, если бы существовал вывод такого вида, длина которого была бы больше числа нетерминалов грамматики $A \Rightarrow B_1\alpha_1 \Rightarrow B_2\alpha_2 \Rightarrow \ldots \Rightarrow B_m\alpha_m = B\alpha$, то какие-то два нетерминала (B_i и B_j при $i \ne j$) неизбежно оказались бы одинаковыми. Другими словами, мы имели бы вывод $B_i\alpha_i \underset{\text{lm}}{\overset{+}{\Rightarrow}} B_j\alpha_j$ при $B_i = B_j$, что означало бы леворекурсивность грамматики.

Поскольку согласно теореме 2.3 $LL(k)$-анализатор аккуратно воспроизводит в своем магазине открытые части сентенциальных форм левостороннего вывода входной цепочки, то участкам вывода вида $B_1\alpha_1 \Rightarrow B_2\alpha_2 \Rightarrow \ldots \Rightarrow B_m\alpha_m$ соответствуют движения типа 1, следующие непосредственно друг за другом, и их число не превосходит c.

Пусть цепочка $x \in L(G)$, где G — $LL(k)$-грамматика и $n = |x|$. Разбирая цепочку x, k-предсказывающий алгоритм анализа, построенный для грамматики G, совершает ровно n pop-движений и не более c движений типа 1 перед каждым из них. Следовательно, общее число движений — не более чем $n + c \times n$. Что и требовалось доказать.

§ 2.6. Тестирование *LL*(*k*)-грамматик

Пусть имеется контекстно-свободная грамматика G. Алгоритмы построения управляющих таблиц $LL(k)$-анализаторов 2.1 и 2.3 достигают успеха только если G — $LL(k)$-грамматика. Поэтому естественно поинтересоваться, является ли данная грамматика G $LL(k)$-грамматикой для данного значения k. Для ответа на этот вопрос можно воспользоваться теоремой 2.1, а для $k = 1$ и сильных $LL(k)$-грамматик также и теоремой 2.2 при данном значении k. Напомним, что в общем случае cfg $G = (V_N, V_T, P, S)$ не является $LL(k)$-грамматикой тогда и только тогда, когда существуют левосторонний вывод вида $S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha$ и пара различных A-правил $A \to \beta$ и $A \to \gamma \in P$ $(\beta \neq \gamma)$, для которых $(\mathrm{FIRST}_k^G(\beta) \oplus_k L) \cap (\mathrm{FIRST}_k^G(\gamma) \oplus_k L) \neq \varnothing$, где $L = \mathrm{FIRST}_k^G(\alpha)$.

Для описания алгоритма, разрешающего этот вопрос, введем вспомогательную функцию.

***Определение* 2.12.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика и $A \in V_N$. Определим $\sigma(A) = \{L \subseteq V_T^{*k} \mid \exists\, S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha, w \in V_T^*$ и $L = \mathrm{FIRST}_k^G(\alpha)\}$.

Алгоритм 2.4: тестирование $LL(k)$-грамматик.

Вход: $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика.

Выход: да, если G — $LL(k)$-грамматика; нет — в противном случае.

Метод.

1. Для каждого нетерминала A, для которого существуют две или более альтернативы, вычисляется $\sigma(A)$.

2. Пусть $A \to \beta$ и $A \to \gamma \in P$ $(\beta \neq \gamma)$ — два различных A-правила. Для каждого $L \in \sigma(A)$ вычисляется $f(L) = (\mathrm{FIRST}_k^G(\beta) \oplus_k L) \cap (\mathrm{FIRST}_k^G(\gamma) \oplus_k L)$. Если $f(L) \neq \varnothing$, то алгоритм завершается с результатом “нет”. Если $f(L) = \varnothing$ для всех $L \in \sigma(A)$, то шаг 2 повторяется для всех пар A-правил.

3. Повторять шаги 1 и 2 для всех нетерминалов из множества V_N.

4. Завершить алгоритм с результатом “да”. Этот шаг выполняется, если были рассмотрены все нетерминалы и алгоритм не завершился на шаге 2.

Разумеется, это описание можно считать алгоритмом, если мы располагаем алгоритмами для вычисления функций $\mathrm{FIRST}_k^G(\beta)$ для $\beta \in V^*$ и $\sigma(A)$ для $A \in V_N$.

§ 2.7. Вычисление функции $\mathbf{FIRST_k^G(\beta)}$

Алгоритм 2.5: вычисление $\mathrm{FIRST}_k^G(\beta)$ для $\beta \in V^*$.

Вход: $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика и $\beta = X_1 X_2 \ldots X_n$, $X_i \in V$ $(i = 1, 2, \ldots, n; n \geq 0)$.

Выход: $\mathrm{FIRST}_k^G(\beta)$.

Метод. Согласно лемме 2.1 $\mathrm{FIRST}_k^G(\beta) = \mathrm{FIRST}_k^G(X_1) \oplus_k \mathrm{FIRST}_k^G(X_2) \oplus_k \ldots \oplus_k \mathrm{FIRST}_k^G(X_n)$, так что задача сводится к вычислению $\mathrm{FIRST}_k^G(X)$ для $X \in V$.

Если $X \in V_T \cup \{\varepsilon\}$, то вычислять нечего, так как в этом случае $\mathrm{FIRST}_k^G(X) = \{X\}$. Остается найти способ вычисления $\mathrm{FIRST}_k^G(X)$ для $X \in V_N$.

Мы будем использовать метод последовательных приближений и строить последовательности множеств $F_i(X)$ для всех $X \in V_N \cup V_T$, $i = 0, 1, 2, \ldots$.

1. $F_i(a) = \{a\}$ для всех $a \in V_T$ и $i \geq 0$.

2. $F_0(A) = \{x \in V_T^{*k} \mid \exists\, A \to x\alpha \in P$, где $|x| = k$, либо $|x| < k$ и $\alpha = \varepsilon\}$.

3. Предположим, что $F_0(A), F_1(A), \ldots, F_{i-1}(A)$ уже построены для всех $A \in V_N$. Тогда

$$F_i(A) = F_{i-1}(A) \cup \{x \in V_T^{*k} \mid \exists\ A \to Y_1 Y_2 \ldots Y_m \in P,\ x \in F_{i-1}(Y_1) \oplus_k F_{i-1}(Y_2) \oplus_k \ldots \oplus_k F_{i-1}(Y_m)\}.$$

4. Так как $F_{i-1}(A) \subseteq F_i(A) \subseteq V_T^{*k}$ для всех $A \in V_N$ и $i > 0$, то шаг 3 требуется повторять до тех пор, пока при некотором $i = j$ не окажется $F_j(A) = F_{j+1}(A)$ для всех $A \in V_N$. Очевидно, что тогда $F_j(A) = F_{j+1}(A) = F_{j+2}(A) = \ldots$.

5. Остается только положить $\mathrm{FIRST}_k^G(A) = F_j(A)$.

Пример 2.10. Рассмотрим еще раз $LL(1)$-грамматику из примера 2.6:
$G = (\{E, E', T, T', F\}, \{a, +, *, (,)\}, P, E)$, где $P = \{(1)\ E \to TE',\ (2)\ E' \to +TE',$
$(3)\ E' \to \varepsilon,\ (4)\ T \to FT',\ (5)\ T' \to *FT',\ (6)\ T' \to \varepsilon,\ (7)\ F \to (E),\ (8)\ F \to a\}$,
и построим функцию $\mathrm{FIRST}_1^G(A)$ для всех $A \in \{E, E', T, T', F\}$.

Прежде всего согласно шагу 2 алгоритма 2.5 получаем

$F_0(E) = F_0(T) = \varnothing$, $F_0(E') = \{+, \varepsilon\}$, $F_0(T') = \{*, \varepsilon\}$, $F_0(F) = \{(, a\}$.

Далее шаг 3 дает

$F_1(E) = F_0(T) \oplus_1 F_0(E') \cup F_0(E) = \varnothing \oplus_1 \{+, \varepsilon\} \cup \varnothing = \varnothing = F_0(E)$;

$F_1(E') = F_0(+) \oplus_1 F_0(T) \oplus_1 F_0(E') \cup F_0(\varepsilon) \cup F_0(E') =$
$= \{+\} \oplus_1 \varnothing \oplus_1 \{+, \varepsilon\} \cup \{\varepsilon\} \cup \{+, \varepsilon\} = \{+, \varepsilon\} = F_0(E')$;

$F_1(T) = F_0(F) \oplus_1 F_0(T') \cup F_0(T) = \{(, a\} \oplus_1 \{*, \varepsilon\} \cup \varnothing = \{(, a\} \neq F_0(T)$;

$F_1(T') = F_0(*) \oplus_1 F_0(F) \oplus_1 F_0(T') \cup F_0(\varepsilon) \cup F_0(T') =$
$= \{*\} \oplus_1 \{(, a\} \oplus_1 \{*, \varepsilon\} \cup \{\varepsilon\} \cup \{*, \varepsilon\} = \{*, \varepsilon\} = F_0(T')$;

$$F_1(F) = F_0(\,(\,) \oplus_1 F_0(E) \oplus_1 F_0(\,)) \cup F_0(a) \cup F_0(F) =$$
$$= \{(\} \oplus_1 \varnothing \oplus_1 \{)\} \cup \{a\} \cup \{(, a\} = \{(, a\} = F_0(F).$$

Поскольку процесс не сошелся для всех нетерминалов, необходимо повторить шаг 3, что дает

$$F_2(E) = F_1(T) \oplus_1 F_1(E') \cup F_1(E) = \{(, a\} \oplus_1 \{+, \varepsilon\} \cup \varnothing = \{(, a\} \neq F_1(E);$$
$$F_2(E') = \{+, \varepsilon\} = F_1(E');$$
$$F_2(T) = \{(, a\} = F_1(T);$$
$$F_2(T') = \{*, \varepsilon\} = F_1(T');$$
$$F_2(F) = \{(, a\} = F_1(F).$$

Так как $F_2(E) \neq F_1(E)$ придется еще раз проделать шаг 3, что дает, наконец,

$$F_3(E) = \{(, a\} = F_2(E);$$
$$F_3(E') = \{+, \varepsilon\} = F_2(E');$$
$$F_3(T) = \{(, a\} = F_2(T);$$
$$F_3(T') = \{*, \varepsilon\} = F_2(T');$$
$$F_3(F) = \{(, a\} = F_2(F).$$

Таким образом, окончательно получаем:

$$\mathrm{FIRST}_1^G(E) = \mathrm{FIRST}_1^G(T) = \mathrm{FIRST}_1^G(F) = \{(, a\};$$
$$\mathrm{FIRST}_1^G(E') = \{+, \varepsilon\};$$
$$\mathrm{FIRST}_1^G(T') = \{*, \varepsilon\}.$$

Теорема 2.7. *Пусть* $G = (V_N, V_T, P, S)$ *— контекстно-свободная грамматика. Алгоритм* 2.5 *правильно вычисляет функцию* $\mathrm{FIRST}_k^G(\beta)$ *для любой цепочки* $\beta \in V^*$ *и* $k \geq 0$.

Доказательство. Фактически достаточно показать, что $F_j(A) = \mathrm{FIRST}_k^G(A)$, если $F_i(A) = F_j(A)$ при $i > j$ для всех $A \in V_N$. Не нарушая общности рассуждений, будем предполагать, что G — приведенная грамматика.

I. Покажем сначала, что $F_j(A) \subseteq \mathrm{FIRST}_k^G(A)$.

Пусть $x \in F_l(A)$ при $l \leq j$. Индукцией по l покажем, что тогда $x \in \mathrm{FIRST}_k^G(A)$.

База. Пусть $l = 0$ и $x \in F_0(A)$. Тогда согласно шагу 2 алгоритма 2.5 существует правило вида $A \to x\alpha \in P$, где $x \in V_T^{*k}$ и $|x| = k$ либо $|x| < k$ и $\alpha = \varepsilon$. Следовательно, $A \overset{*}{\Rightarrow} xy$, где $|x| = k$ и $y \in V_T^*$ либо $|x| < k$ и $y = \varepsilon$. Согласно определению функции FIRST_k^G заключаем, что $x \in \mathrm{FIRST}_k^G(A)$. База доказана.

Индукционная гипотеза. Предположим, что аналогичное утверждение выполняется для всех $l \leq n$ $(0 \leq n \leq j)$.

Индукционный переход. Докажем, что тогда аналогичное утверждение верно для $l = n + 1$.

Пусть $x \in F_{n+1}(A)$. Согласно алгоритму 2.5 либо $x \in F_n(A)$ и тогда в соответствии с индукционным предположением $x \in \mathrm{FIRST}_k^G(A)$ либо существует правило вида $A \to Y_1 Y_2 \ldots Y_m \in P$ и $x \in F_n(Y_1) \oplus_k F_n(Y_2) \oplus_k \ldots \oplus_k F_n(Y_m)$. Если $Y_i \in V_\mathrm{N}$, то в соответствии с индукционной гипотезой $F_n(Y_i) \subseteq \mathrm{FIRST}_k^G(Y_i)$. Если же $Y_i \in V_\mathrm{T}$, то согласно определению множеств F_n и функции FIRST_k^G заключаем, что $F_n(Y_i) = \{Y_i\} = \mathrm{FIRST}_k^G(Y_i)$.

Итак, в любом случае справедливо $F_n(Y_i) \subseteq \mathrm{FIRST}_k^G(Y_i)$ для $Y_i \in V_\mathrm{N} \cup V_\mathrm{T}$. Тогда

$$x \in F_n(Y_1) \oplus_k F_n(Y_2) \oplus_k \ldots \oplus_k F_n(Y_m) \subseteq$$
$$\subseteq \mathrm{FIRST}_k^G(Y_1) \oplus_k \mathrm{FIRST}_k^G(Y_2) \oplus_k \ldots \oplus_k \mathrm{FIRST}_k^G(Y_m) = \mathrm{FIRST}_k^G(Y_1 Y_2 \ldots Y_m) \subseteq$$
$$\subseteq \mathrm{FIRST}_k^G(A),$$

поскольку $A \to Y_1 Y_2 \ldots Y_m \in P$. Что и требовалось доказать.

II. Покажем теперь, что $\mathrm{FIRST}_k^G(A) \subseteq F_j(A)$.

Пусть $x \in \mathrm{FIRST}_k^G(A)$. По определению этой функции существует вывод $A \overset{*}{\Rightarrow} xy$, где $|x| = k$ и $y \in V_\mathrm{T}^*$ либо $|x| < k$ и $y = \varepsilon$. Индукцией по длине вывода l покажем, что если $x \in \mathrm{FIRST}_k^G(A)$ благодаря существованию такого вывода, то $x \in F_j(A)$.

База. Пусть $l = 1$. Имеем $A \Rightarrow xy$, где $|x| = k$ и $y \in V_\mathrm{T}^*$ либо $|x| < k$ и $y = \varepsilon$. Тогда существует правило $A \to xy \in P$ и согласно шагу 2 алгоритма 2.5 $x \in F_0(A) \subseteq F_j(A)$. База доказана.

Индукционная гипотеза. Предположим, что аналогичное утверждение выполняется для всех $l \le n$ $(n \ge 1)$.

Индукционный переход. Докажем, что тогда аналогичное утверждение верно для $l = n + 1$.

Пусть $A \Rightarrow Y_1 Y_2 \ldots Y_m \overset{n}{\Rightarrow} y_1 y_2 \ldots y_m$ — вывод длиной $n + 1$, где $y_i \in V_\mathrm{T}^*$, $Y_i \overset{n_i}{\Rightarrow} y_i$, $n_i \le n$, $i = 1, 2, \ldots, n$ и $x = \mathrm{FIRST}_k^G(y_1 y_2 \ldots y_m) \in \mathrm{FIRST}_k^G(A)$. Ясно, что

$$x = \mathrm{FIRST}_k^G(y_1 y_2 \ldots y_m) = \mathrm{FIRST}_k^G(y_1) \oplus_k \mathrm{FIRST}_k^G(y_2) \oplus_k \ldots \oplus_k \mathrm{FIRST}_k^G(y_m) \subseteq$$
$$\subseteq \mathrm{FIRST}_k^G(Y_1) \oplus_k \mathrm{FIRST}_k^G(Y_2) \oplus_k \ldots \oplus_k \mathrm{FIRST}_k^G(Y_m) \subseteq$$
$$\subseteq F_n(Y_1) \oplus_k F_n(Y_2) \oplus_k \ldots \oplus_k F_n(Y_m).$$

Последнее вложение множеств имеет место в соответствии со следствием из индукционной гипотезы, примененной к выводам $Y_i \overset{n_i}{\Rightarrow} y_i$, $n_i \le n$, с учетом того, что при $n_i = 0$ имеем $Y_i = y_i$.

Итак, существует правило $A \to Y_1 Y_2 \ldots Y_m \in P$ и $x \in F_n(Y_1) \oplus_k F_n(Y_2) \oplus_k \ldots \oplus_k F_n(Y_m)$. Согласно шагу 3 алгоритма 2.5 заключаем, что $x \in F_{n+1}(A) \subseteq F_j(A)$. Что и требовалось доказать.

Из рассуждений I и II следует равенство $F_j(A) = \mathrm{FIRST}_k^G(A)$, что равнозначно утверждению теоремы.

§ 2.8. Вычисление функции σ(*A*)

Обратимся теперь к описанию алгоритма вычисления функции σ(*A*).

Алгоритм 2.6: вычисление σ(*A*) для $A \in V_N$.

Вход: $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика.

Выход: σ(*A*) для всех $A \in V_N$.

Метод.

По определению $\sigma(A) = \{L \subseteq V_T^{*k} \mid \exists\, S \underset{lm}{\overset{*}{\Rightarrow}} wA\alpha, w \in V_T^* \text{ и } L = \mathrm{FIRST}_k^G(\alpha)\}$.

Введем в рассмотрение еще одну вспомогательную функцию, определяемую для пары нетерминалов следующим образом:

$$\sigma'(A, B) = \{L \subseteq V_T^{*k} \mid \exists\, A \underset{lm}{\overset{*}{\Rightarrow}} wB\alpha,\ w \in V_T^* \text{ и } L = \mathrm{FIRST}_k^G(\alpha)\}.$$

Очевидно, что $\sigma(A) = \sigma'(S, A)$. Таким образом, достаточно дать алгоритм вычисления функции $\sigma'(A, B)$ для любых $A, B \in V_N$. Мы будем вычислять $\sigma'(A, B)$ методом последовательных приближений, параллельно выстраивая последовательности множеств $\sigma_i'(A, B)$ для всех возможных пар $(A, B) \in V_N \times V_N$ следующим образом:

1. $\sigma_0'(A, B) = \{L \subseteq V_T^{*k} \mid \exists\, A \to \beta B\alpha \in P,\ \beta, \alpha \in V_N^*,\ L = \mathrm{FIRST}_k^G(\alpha)\}$.

2. Пусть $\sigma_0'(A, B), \sigma_1'(A, B), \ldots, \sigma_i'(A, B)$ вычислены для всех пар нетерминалов. Положим

$$\sigma_{i+1}'(A,B) = \sigma_i'(A,B) \cup \{L \subseteq V_T^{*k} \mid \exists\, A \to X_1X_2\ldots X_m \in P, L' \in \sigma_i'(X_p, B),\ X_p \in V_N,\ 1 \le p \le m,\ L = L' \oplus_k \mathrm{FIRST}_k^G(X_{p+1}X_{p+2}\ldots X_m)\}.$$

3. Повторять шаг 2 до тех пор, пока при некотором $i = j$ не окажется, что $\sigma_{j+1}'(A,B) = \sigma_j'(A,B)$ для всех $(A,B) \in V_N \times V_N$. Такое j существует, потому что $\sigma_0'(A,B) \subseteq \sigma_1'(A,B) \subseteq \ldots \subseteq 2^{V_T^{*k}}$.

4. Полагаем $\sigma'(A,B) = \sigma_j'(A,B)$.

5. Наконец, $\sigma(A) = \sigma'(S,A)$. Если окажется, в частности, что $\{\varepsilon\} \notin \sigma'(S, S)$, то $\sigma(S) = \sigma'(S, S) \cup \{\{\varepsilon\}\}$.

Пример 2.11. Пусть $G = (\{S, A\}, \{a, b\}, P, S)$, где $P = \{S \to AS, S \to \varepsilon, A \to aA, A \to b\}$.

Табл. 2.5

σ_i'	$i = 0$	$i = 1$
(S, S)	$\{\{\varepsilon\}\}$	$\{\{\varepsilon\}\}$
(S, A)	$\{\{\varepsilon, a, b\}\}$	$\{\{\varepsilon, a, b\}\}$
(A, S)	$\varnothing$	$\varnothing$
(A, A)	$\{\{\varepsilon\}\}$	$\{\{\varepsilon\}\}$

Вычислим функции $\sigma(S)$ и $\sigma(A)$ при $k = 1$. Сначала получаем $\mathrm{FIRST}_1^G(S) = \{\varepsilon, a, b\}$ и $\mathrm{FIRST}_1^G(A) = \{a, b\}$. Затем построим последовательности $\sigma_i'(X, Y)$, где $(X, Y) \in V_N \times V_N$ (табл. 2.5). За два шага получаем $\sigma(S) = \{\{\varepsilon\}\}$, $\sigma(A) = \{\{\varepsilon, a, b\}\}$.

Поясним построение этих последовательностей.

Шаг 1. Построение $\sigma_0'(S, S)$: имеем правило для $S \to AS$, в котором нетерминал S встречается слева и справа. Вычисляется $\mathrm{FIRST}_1^G(\varepsilon) = \{\varepsilon\}$, поскольку правый контекст для S есть ε. Другого правила для S, в котором справа встречается нетерминал S, не существует. Поэтому $\sigma_0'(S, S) = \{\{\varepsilon\}\}$.

Построение $\sigma_0'(S, A)$: существует только одно правило с нетерминалом S в левой части и нетерминалом A — в правой. Это правило $S \to AS$. Вычисляется FIRST_1^G от правого контекста A, т.е. $\mathrm{FIRST}_1^G(S) = \{\varepsilon, a, b\}$. Соответственно $\sigma_0'(S, A) = \{\{\varepsilon, a, b\}\}$.

Построение $\sigma_0'(A, S)$: не существует ни одного правила с нетерминалом A в левой части и нетерминалом S — в правой. Поэтому $\sigma_0'(A, S) = \varnothing$.

Построение $\sigma_0'(A, A)$: имеется только одно продуктивное правило $A \to aA$, дающее по пустому правому контексту A значение $\sigma_0'(A, A) = \{\{\varepsilon\}\}$.

Шаг 2. Построение $\sigma_1'(S, S)$: имеем правило для $S \to AS$, в котором справа два нетерминала — два источника для пополнения множества $\sigma_0'(S, S)$ новыми членами. Именно: можно вычислить новый член по первому вхождению нетерминала A в правую часть этого правила и по второму вхождению нетерминала S:

$L = L' \oplus_1 \mathrm{FIRST}_1^G(S)$, где $L' \in \sigma_0'(A, S) = \varnothing$.

Ясно, что из этого источника никакого пополнения не получится. Попробуем воспользоваться другим источником для получения нового члена:

$L = L' \oplus_1 \mathrm{FIRST}_1^G(\varepsilon)$, где $L' \in \sigma_0'(S, S) = \{\{\varepsilon\}\}$.

Существует единственное множество $L' = \{\varepsilon\}$, с помощью которого вычисляется новый член

$$L = L' \oplus_1 \mathrm{FIRST}_1^G(\varepsilon) = \{\varepsilon\} \oplus_1 \{\varepsilon\} = \{\varepsilon\},$$

но такое множество уже есть в $\sigma_0'(S, S)$.

На рис. 2.2 представлена схема соответствия параметров, используемых при вычислении множества $\sigma_1'(S, S)$.

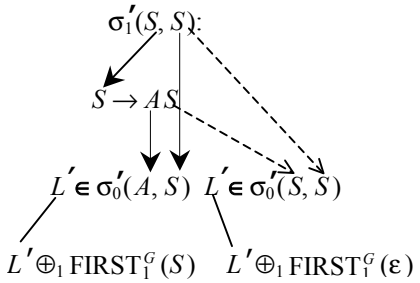

Рис. 2.2.

Аналогичным образом пополняются множества $\sigma_0'(S,A)$, $\sigma_0'(A,S)$ и $\sigma_0'(A,A)$. Однако добавить к этим множествам новые члены, как нетрудно проверить, не удается.

Теперь все готово для тестирования данной cfg G на предмет ее принадлежности к классу $LL(1)$-грамматик. Имеем два альтернативных правила: $S \to AS$, $S \to \varepsilon$ для нетерминала S. Требуется вычислить

$(\mathrm{FIRST}_1^G(AS) \oplus_1 L) \cap (\mathrm{FIRST}_1^G(\varepsilon) \oplus_1 L)$, где $L \in \sigma(S) = \{\{\varepsilon\}\}$.

Получаем

$$(\mathrm{FIRST}_1^G(AS) \oplus_1 L) = \{a, b\} \oplus_1 \{\varepsilon\} = \{a, b\},$$
$$(\mathrm{FIRST}_1^G(\varepsilon) \oplus_1 L) = \{\varepsilon\} \oplus_1 \{\varepsilon\} = \{\varepsilon\}$$

и тогда

$$(\mathrm{FIRST}_1^G(AS) \oplus_1 L) \cap (\mathrm{FIRST}_1^G(\varepsilon) \oplus_1 L) = \{a, b\} \cap \{\varepsilon\} = \varnothing.$$

Имеем также два альтернативных правила $A \to aA$, $A \to b$ для нетерминала A. Требуется вычислить $(\mathrm{FIRST}_1^G(aA) \oplus_1 L) \cap (\mathrm{FIRST}_1^G(b) \oplus_1 L)$, где $L \in \sigma(A) = \{\{\varepsilon, a, b\}\}$. Получаем

$$(\mathrm{FIRST}_1^G(aA) \oplus_1 L) = \{a\} \oplus_1 \{\varepsilon, a, b\} = \{a\},$$
$$(\mathrm{FIRST}_1^G(b) \oplus_1 L) = \{b\} \oplus_1 \{\varepsilon\} = \{b\}$$

и тогда

$$(\mathrm{FIRST}_1^G(aS) \oplus_1 L) \cap (\mathrm{FIRST}_1^G(b) \oplus_1 L) = \{a\} \cap \{b\} = \varnothing.$$

Так как оба пересечения пусты, то данная грамматика G действительно $LL(1)$-грамматика.

Теорема 2.8. Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. Алгоритм 2.6 правильно вычисляет функцию $\sigma(A)$ для любого нетерминала $\sigma \in V_N$ и $k \geq 0$.

Доказательство. Фактически достаточно показать, что $\sigma_j'(A, B) = \sigma'(A, B)$, если $\sigma_i'(A, B) = \sigma_j'(A, B)$ при $i > j$ для всех $A, B \in V_N$. Не нарушая общности рассуждений, будем предполагать, что G — приведенная грамматика.

I. Покажем сначала, что $\sigma_j'(A, B) \subseteq \sigma'(A, B)$.

Индукцией по i покажем, что $\sigma_i'(A, B) \subseteq \sigma'(A, B)$ для любого $i \geq 0$.

База. Пусть $i = 0$, т.е. $L \in \sigma_0'(A, B)$. Согласно шагу 1 алгоритма 2.6 существует правило $A \to \beta B \alpha \in P$, $\beta, \alpha \in V^*$, $L = \mathrm{FIRST}_k^G(\alpha)$. Поскольку грамматика G приведенная, то существует вывод, в частности левосторонний, $A \underset{\text{lm}}{\overset{*}{\Longrightarrow}} wB\alpha$. Следовательно, по определению $L = \mathrm{FIRST}_k^G(\alpha) \in \sigma'(A, B)$. База доказана.

Индукционная гипотеза. Предположим, что аналогичное утверждение выполняется для всех $i \leq n$ $(0 \leq n \leq j)$.

Индукционный переход. Докажем, что тогда аналогичное утверждение верно для $i = n + 1$.

Пусть $L \in \sigma'_{n+1}(A, B)$. Согласно шагу 2 алгоритма 2.6 либо $L \in \sigma'_n(A, B)$ и тогда в соответствии с индукционной гипотезой $L \in \sigma'(A, B)$, либо существуют правило $A \to X_1X_2 \ldots X_m \in P$ и $L' \in \sigma'_n(X_p, B)$, $X_p \in V_N$, $1 \le p \le m$, такие, что $L = L' \oplus_k \mathrm{FIRST}_k^G(X_{p+1}X_{p+2} \ldots X_m)$. Согласно индукционной гипотезе, примененной к $L' \in \sigma'_n(X_p, B)$, можем утверждать, что $L' \in \sigma'(X_p, B)$ и что существует левосторонний вывод вида $X_p \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_pB\gamma$, а $L' = \mathrm{FIRST}_k^G(\gamma)$. Тогда можно построить левосторонний вывод:

$$A \overset{*}{\underset{\text{lm}}{\Rightarrow}} X_1X_2 \ldots X_{p-1}X_pX_{p+1} \ldots X_m \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_1w_2 \ldots w_{p-1}X_pX_{p+1} \ldots X_m \overset{*}{\underset{\text{lm}}{\Rightarrow}}$$
$$\overset{*}{\underset{\text{lm}}{\Rightarrow}} w_1w_2 \ldots w_{p-1}w_pB\gamma X_{p+1} \ldots X_m.$$

Согласно определению $\mathrm{FIRST}_k^G(\gamma X_{p+1} \ldots X_m) \in \sigma'(A, B)$. Кроме того,

$$\mathrm{FIRST}_k^G(\gamma X_{p+1} \ldots X_m) = \mathrm{FIRST}_k^G(\gamma) \oplus_k \mathrm{FIRST}_k^G(X_{p+1} \ldots X_m) =$$
$$= L' \oplus_k \mathrm{FIRST}_k^G(X_{p+1} \ldots X_m) = L.$$

Итак, $L \in \sigma'(A, B)$. Что и требовалось.

II. Покажем теперь, что $\sigma'(A, B) \subseteq \sigma'_j(A, B)$.

Пусть $L \in \sigma'(A,B)$. Это значит, что существует левосторонний вывод $A \overset{l}{\underset{\text{lm}}{\Rightarrow}} wB\alpha$, $w \in V_T^*$ и $L = \mathrm{FIRST}_k^G(\alpha)$. Индукцией по длине вывода l покажем, что $L \in \sigma'_j(A, B)$.

База. Пусть $l = 1$, т.е. $A \underset{\text{lm}}{\Rightarrow} wB\alpha, w \in V_T^*$ и $L = \mathrm{FIRST}_k^G(\alpha)$. Тогда существует правило $A \to wB\alpha$ и согласно шагу 1 алгоритма 2.6 $L = \mathrm{FIRST}_k^G(\alpha) \in \sigma'_0(A, B) \subseteq \sigma'_j(A, B)$. База доказана.

Индукционная гипотеза. Предположим, что аналогичное утверждение выполняется для всех $l \le n$ $(n \ge 1)$.

Индукционный переход. Докажем, что тогда аналогичное утверждение верно для $l = n + 1$. Пусть

$$A \underset{\text{lm}}{\Rightarrow} X_1X_2 \ldots X_{p-1}X_pX_{p+1} \ldots X_m \overset{n-1}{\underset{\text{lm}}{\Rightarrow}} w_1w_2 \ldots w_{p-1}X_pX_{p+1} \ldots X_m \underset{\text{lm}}{\Rightarrow}$$
$$\underset{\text{lm}}{\Rightarrow} w_1w_2 \ldots w_{p-1}w_pB\gamma X_{p+1} \ldots X_m = wB\alpha$$

— вывод длиной $n + 1$, где $w = w_1w_2 \ldots w_p$, $X_p \to w_pB\gamma \in P$, $\alpha = \gamma X_{p+1} \ldots X_m$, и

$$L = \mathrm{FIRST}_k^G(\alpha) = \mathrm{FIRST}_k^G(\gamma X_{p+1} \ldots X_m) = L' \oplus_k \mathrm{FIRST}_k^G(X_{p+1} \ldots X_m),$$

где $L' = \mathrm{FIRST}_k^G(\gamma)$. Именно благодаря этому обстоятельству $L \in \sigma'(A, B)$.

Одновременно согласно шагу 2 алгоритма 2.6 из существования вывода $X_p \overset{l_p}{\underset{\text{lm}}{\Rightarrow}} w_pB\gamma$, $l_p \le n$, следует по определению, что $L' = \mathrm{FIRST}_k^G(\gamma) \in \sigma'(X_p, B)$, а по ин-

дукционному предположению, что $\sigma'(X_p, B) \subseteq \sigma'_j(X_p, B)$ и, следовательно, $L' \in \sigma'_j(X_p, B)$. Кроме того, имеется правило $A \to X_1X_2\ldots X_pX_{p+1}\ldots X_m \in P$. Согласно шагу 2 алгоритма 2.6 $L = L' \oplus_k \mathrm{FIRST}_k^G(X_{p+1}\ldots X_m)$, где $L' \in \sigma'_j(X_p, B)$ есть элемент $\sigma'_{j+1}(A, B)$. Но $\sigma'_{j+1}(A, B) = \sigma'_j(A, B)$. Итак, $L \in \sigma'_j(A, B)$.

Из рассуждений I и II следует равенство $\sigma'_j(A, B) = \sigma'(A, B)$ и утверждение теоремы.

§ 2.9. Алгоритм вычисления функции $\mathrm{FOLLOW}_k^G(A)$

Сопоставляя определения функций

$$\sigma(A) = \{L \subseteq V_T^{*k} \mid \exists\, S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha, w \in V_\mathrm{T}^*,\ L = \mathrm{FIRST}_k^G(\alpha)\}$$

и

$$\mathrm{FOLLOW}_k^G(A) = \{w \in V_\mathrm{T}^* \mid \exists\, S \overset{*}{\underset{G}{\Rightarrow}} \gamma A\alpha \text{ и } w \in \mathrm{FIRST}_k^G(\alpha)\},\ A \in V_\mathrm{N},$$

нетрудно сообразить, что

$$\mathrm{FOLLOW}_k^G(A) = \bigcup_{L \in \sigma(A)} L\ .$$

Это равенство — ключ к модификации алгоритма вычисления $\sigma(\mathrm{A})$, дающей алгоритм вычисления $\mathrm{FOLLOW}_k^G(A)$.

Алгоритм 2.7: вычисление $\mathrm{FOLLOW}_k^G(A)$.

Вход: $G = (V_\mathrm{N}, V_\mathrm{T}, P, S)$ — контекстно-свободная грамматика.

Выход: $\mathrm{FOLLOW}_k^G(A)$ для всех $A \in V_\mathrm{N}$.

Метод.

Определим вспомогательную функцию $\varphi(A, B) = \{w \in \Sigma^{*k} \mid \exists\, A \overset{*}{\underset{G}{\Rightarrow}} \gamma B\alpha,\ w \in \mathrm{FIRST}_k^G(\alpha)\}$, где $A, B \in V_\mathrm{N}$. Очевидно, что $\mathrm{FOLLOW}_k^G(A) = \varphi(S, A)$. Остается определить алгоритм для вычисления функции $\varphi(A, B)$.

Вычисление значения $\varphi(A, B)$ производится по следующим шагам:

1. $\varphi_0(A, B) = \{\, w \in \Sigma^{*k} \mid \exists\, A \to \gamma B\alpha \in P,\ \alpha, \gamma \in V^*,\ w = \mathrm{FIRST}_k^G(\alpha)\}$.

2. Пусть значения $\varphi_0(A, B)$, $\varphi_1(A, B), \ldots,$ $\varphi_i(A, B)$ уже построены для всех $(A, B) \in V_\mathrm{N} \times V_\mathrm{N}$. Тогда

$$\varphi_{i+1}(A, B) = \varphi_i(A, B) \cup \{w \in \Sigma^{*k} \mid \exists\, A \to X_1X_2\ldots X_pX_{p+1}\ldots X_m \in P, X_p \in V_\mathrm{N},\ w \in \varphi_i(X_p, B) \oplus_k \mathrm{FIRST}_k^G(X_{p+1}\ldots X_m)\}.$$

3. Шаг 2 повторяется до тех пор, пока при некотором $i = j$ не окажется $\varphi_{j+1}(A, B) = \varphi_j(A, B)$ для всех $(A, B) \in V_\mathrm{N} \times V_\mathrm{N}$. Такое j существует, ибо $\varphi_0(A, B) \subseteq \varphi_1(A, B) \subseteq \ldots \subseteq \Sigma^{*k}$ при том, что последнее множество является конечным. И тогда $\varphi(A, B) = \varphi_j(A, B)$.

4. Полагаем $\mathrm{FOLLOW}_k^G(A) = \varphi(S, A)$ для всех $A \in V_\mathrm{N} \setminus \{S\}$ и $\mathrm{FOLLOW}_k^G(S) = \varphi(S,S) \cup \{\varepsilon\}$, так как всегда $\varepsilon \in \mathrm{FOLLOW}_k^G(S)$, но может оказаться, что $\varepsilon \notin \varphi(S,S)$.

Пример 2.12. Пусть $G = (\{S, A\}, \{a, b\}, P, S)$, где $P = \{S \to aAaa, S \to bAba, A \to b, A \to \varepsilon\}$.

Построим множества $\mathrm{FOLLOW}_2^G(S)$ и $\mathrm{FOLLOW}_2^G(A)$, используя описанный алгоритм (табл. 2.6).

Табл. 2.6

$N \times N$	φ_0	φ_1
(S, S)	$\varnothing$	$\varnothing$
(S, A)	$\{aa, ba\}$	$\{aa, ba\}$
(A, S)	$\varnothing$	$\varnothing$
(A, A)	$\varnothing$	$\varnothing$

Получаем

$$\mathrm{FOLLOW}_2^G(S) = \varphi_0(S, S) \cup \{\varepsilon\} = \{\varepsilon\};\ \mathrm{FOLLOW}_2^G(A) = \varphi_0(S, A) = \{aa, ba\}.$$

Покажем теперь, что алгоритм 2.7 действительно вычисляет функцию $\mathrm{FOLLOW}_k^G(A)$ для любого нетерминала A.

Теорема 2.9. *Пусть* $G = (V_\mathrm{N}, V_\mathrm{T}, P, S)$ *— контекстно-свободная грамматика. Алгоритм* 2.7 *правильно вычисляет функцию* $\mathrm{FOLLOW}_k^G(A)$ *для любого* $A \in V_\mathrm{N}$ *и* $k \geq 0$.

Доказательство. Фактически достаточно показать, что $\varphi_j(A, B) = \varphi(A,B)$, если $\varphi_i(A, B) = \varphi_j(A, B)$ при $i > j$ для всех $A, B \in V_\mathrm{N}$.

I. Покажем сначала, что $\varphi_j(A, B) \subseteq \varphi(A,B)$ или, что то же самое, если $w \in \varphi_l(A, B)$ при $l \leq j$, то $w \in \varphi(A, B)$, используя индукцию по l.

База. Пусть $l = 0$. Имеем $w \in \varphi_0(A, B)$. Согласно шагу 1 алгоритма 2.7 существует правило $A \to \gamma B \alpha \in P$, $\alpha, \gamma \in V^*$, и $w \in \mathrm{FIRST}_k^G(\alpha)$. Это правило позволяет построить вывод $A \underset{G}{\Rightarrow} \gamma B \alpha$, причем любая цепочка $w \in \mathrm{FIRST}_k^G(\alpha)$ также является элементом множества $\varphi(A,B)$ — таково определение этой функции, т.е. $w \in \varphi(A,B)$. База доказана.

Индукционная гипотеза. Предположим, что аналогичное утверждение выполняется для всех $l \leq n$ $(0 \leq n \leq j)$.

Индукционный переход. Докажем, что тогда аналогичное утверждение верно для $l = n + 1$.

Итак, пусть $w \in \varphi_{n+1}(A, B)$. Согласно шагу 2 алгоритма 2.7 либо $w \in \varphi_n(A, B)$ и тогда согласно индукционной гипотезе $w \in \varphi(A,B)$, либо существует правило $A \to X_1 X_2 \ldots X_p X_{p+1} \ldots X_m \in P$ и $w \in \varphi_n(X_p, B) \oplus_k \mathrm{FIRST}_k^G(X_{p+1} \ldots X_m)$, где $X_p \in V_\mathrm{N}$,

$1 \le p \le m$. Иначе говоря, по определению операции $\oplus_k$ существуют $w_1 \in \varphi_n(X_p, B)$, $w_2 \in \mathrm{FIRST}_k^G(X_{p+1}\ldots X_m)$ и $w = \mathrm{FIRST}_k^G(w_1w_2)$. В соответствии с индукционным предположением из того, что $w_1 \in \varphi_n(X_p, B)$, следует, что $w_1 \in \varphi(X_p, B)$. Это значит, что $X_p \overset{*}{\underset{G}{\Rightarrow}} \gamma B\alpha$ и $w_1 \in \mathrm{FIRST}_k^G(\alpha)$. Кроме того,

$$w = \mathrm{FIRST}_k^G(w_1w_2) = \mathrm{FIRST}_k^G(w_1) \oplus_k \mathrm{FIRST}_k^G(w_2) \subseteq$$

$$\subseteq \mathrm{FIRST}_k^G(\alpha) \oplus_k \mathrm{FIRST}_k^G(X_{p+1}\ldots X_m) = \mathrm{FIRST}_k^G(\alpha X_{p+1}\ldots X_m)$$

и при этом существует вывод $A \underset{G}{\Rightarrow} X_1X_2\ldots X_pX_{p+1}\ldots X_m \overset{*}{\underset{G}{\Rightarrow}} X_1X_2\ldots\gamma B\alpha X_{p+1}\ldots X_m$. Следовательно, $w \in \varphi(A, B)$.

II. Покажем теперь, что $\varphi(A, B) \subseteq \varphi_j(A, B)$. Пусть $w \in \varphi(A, B)$ благодаря тому, что существует вывод $A \overset{l}{\Rightarrow} \gamma B\alpha$ и $w \in \mathrm{FIRST}_k^G(\alpha)$. Индукцией по длине вывода l покажем, что $w \in \varphi_j(A, B)$.

База. Пусть $l = 1$. Имеем $A \underset{G}{\Rightarrow} \gamma B\alpha$ и $w \in \mathrm{FIRST}_k^G(\alpha)$. Тогда существует правило $A \to \gamma B\alpha \in P$ и согласно шагу 1 алгоритма 2.7 $w \in \varphi_0(A, B) \subseteq \varphi_j(A, B)$. База доказана.

Индукционная гипотеза. Предположим, что аналогичное утверждение выполняется для всех $l \le n$ ($n \ge 1$).

Индукционный переход. Докажем, что тогда аналогичное утверждение верно для $l = n + 1$. Пусть $w \in \varphi(A, B)$ благодаря тому, что существует вывод длиной $n + 1$ вида

$$A \underset{G}{\Rightarrow} X_1X_2\ldots X_{p-1}X_pX_{p+1}\ldots X_m \overset{*}{\underset{G}{\Rightarrow}} \delta X_p\eta \overset{*}{\underset{G}{\Rightarrow}} \underbrace{\delta\beta}_{\gamma} B \underbrace{\chi\eta}_{\alpha} = \gamma B\alpha,$$

в котором

$$X_1X_2\ldots X_{p-1} \overset{*}{\underset{G}{\Rightarrow}} \delta, \quad X_{p+1}\ldots X_m \overset{*}{\underset{G}{\Rightarrow}} \eta, \quad X_p \overset{*}{\underset{G}{\Rightarrow}} \beta B\chi, \quad \alpha = \chi\eta, \quad \gamma = \delta\beta,$$

$$w \in \mathrm{FIRST}_k^G(\alpha) = \mathrm{FIRST}_k^G(\chi\eta) = \mathrm{FIRST}_k^G(\chi) \oplus_k \mathrm{FIRST}_k^G(\eta) \subseteq$$

$$\subseteq \varphi(X_p, B) \oplus_k \mathrm{FIRST}_k^G(X_{p+1}\ldots X_m) \subseteq \varphi_j(X_p, B) \oplus_k \mathrm{FIRST}_k^G(X_{p+1}\ldots X_m) \subseteq$$

$$\subseteq \varphi_{j+1}(A, B) = \varphi_j(A, B),$$

поскольку $\mathrm{FIRST}_k^G(\chi) \subseteq \varphi(X_p, B) \subseteq \varphi_j(X_p, B)$. Последнее вложение следует в соответствии с индукционной гипотезой из существования вывода $X_p \overset{*}{\underset{G}{\Rightarrow}} \beta B\chi$ длиной не больше n.

Здесь использовано существование правила $A \to X_1X_2\ldots X_pX_{p+1}\ldots X_m \in P$, благодаря которому, учитывая шаг 2 алгоритма 2.7, мы заключили, что $w \in \varphi_{j+1}(A, B)$. Что и требовалось доказать.

Из рассуждений I и II следует равенство $\varphi_j(A, B) = \varphi(A, B)$ и утверждение теоремы.

§ 2.10. *k*-Предсказывающий алгоритм трансляции

Ранее в этой части пособия была доказана теорема 1.3 о том, что выходную цепочку простой семантически однозначной трансляции можно сгенерировать по левостороннему анализу входной цепочки посредством детерминированного магазинного преобразователя. Если входная грамматика схемы синтаксически управляемой трансляции принадлежит классу $LL(k)$, то мы имеем детерминированный механизм, правда, не преобразователь, а k-предсказывающий алгоритм анализа, который выдает левосторонний анализ входной цепочки трансляции, задаваемой такой схемой. Кроме того, в лемме 1.1 этой части был описан способ построения недетерминированного магазинного преобразователя, реализующего трансляцию, определяемую простой схемой. Использованный в ней прием можно применить для модификации k-предсказывающего алгоритма анализа в k-предсказывающий алгоритм трансляции, реализующий трансляцию, специфицируемую простой семантически однозначной sdts с входной грамматикой класса $LL(k)$. Это устройство отличается от $LL(k)$-анализатора лишь тем, что имеет еще один тип движений, состоящий в том, чтобы перенести верхний символ магазина, являющийся дубликатом выходного символа, на выходную ленту, превратив его в настоящий выходной символ (как в лемме 1.1 этой части). Детали прояснятся из описания алгоритма 2.8.

Алгоритм 2.8: построение k-предсказывающего алгоритма трансляции.

Вход: $T = (N, \Sigma, \Delta, R, S)$ — простая семантически однозначная схема синтаксически управляемой трансляции с входной грамматикой G_i класса $LL(k)$.

Выход: $\Im$ — k-предсказывающий алгоритм трансляции, реализующий трансляцию $\tau(T)$.

Метод.

1. Предполагая, что множество $\mathcal{T}$ $LL(k)$-таблиц, необходимых для анализа в грамматике G_i, уже построено, положим $\Im = (\Sigma, \Gamma \cup \{\$\}, \Delta, M, X_0, \$)$, где Σ и Δ — такие же, как в схеме T; $\Gamma = \mathcal{T} \cup \Sigma \cup \Delta'$; $\Delta' = \{b' \mid b' = h(b),\ b \in \Delta\}$; $\Sigma \cap \Delta' = \varnothing$; $X_0 = T_0 = T_{S,\{\varepsilon\}}$ — начальный символ магазина; \$ — маркер "дна" магазина; M: $(\Gamma \cup \{\$\}) \times \Sigma^{*k} \to (\Sigma \cup \Delta')^* \cup \{\text{pop, pass, accept, error}\}$ — управляющая таблица, которая строится по следующим шагам:

2. $M(T_{A,L}, u) = x_0 y_0' T_{A_1,L_1} x_1 y_1' T_{A_2,L_2} x_2 y_2' T_{A_3,L_3} \ldots T_{A_m,L_m} x_m y_m'$, если $T_{A,L}(u) = (A \to x_0 A_1 x_1 A_2 x_2 A_3 \ldots A_m x_m, \langle Y_1, Y_2, Y_3, \ldots, Y_m \rangle)$, и $A \to x_0 A_1 x_1 A_2 x_2 A_3 \ldots A_m x_m, y_0 A_1 y_1 A_2\ y_2 A_3 \ldots A_m y_m \in R$ — i-е правило схемы. Здесь $y_i' = h(y_i)$, $i = 0, 1, 2, \ldots, m$.

3. $M(a, u) = \text{pop}$, если $a \in \Sigma$, $u = av$, $v \in \Sigma^{*k-1}$.

4. $M(b', u) = \text{pass}$ для всех $u \in \Sigma^{*k}$. Такой управляющий элемент определяет переход между конфигурациями: $(x, b'\alpha\$, y) \vdash (x, \alpha\$, yb)$.

5. $M(\$, \varepsilon) = \text{accept}$.

6. $M(X, u) = \text{error}$ для всех $(X, u) \in (\Gamma \cup \{\$\}) \times \Sigma^{*k}$, для которых элементы M не определены по пп. 2–5.

Пример 2.13. Пусть $T = (\{S, A\}, \{a, b\}, \{a, b, e, \langle, \rangle\}, R, S)$, где $R = \{(1)\ S \to aAaa, aAaa; (2)\ S \to bAba, \langle A\rangle a; (3)\ A \to b, b; (4)\ A \to \varepsilon, e\}$.

Входная грамматика этой схемы — не сильная $LL(2)$-грамматика, рассматривавшаяся в предыдущих примерах. В примере 2.9 для нее был построен 2-предсказывающий алгоритм анализа. Применяя алгоритм 2.8 к данной схеме, получаем следующий 2-предсказывающий алгоритм трансляции, реализующий определяемую ею трансляцию:

$\Im = (\{a, b\}, \{T_0, T_1, T_2, a, b, a', b', e, \langle, \rangle, \$\}, \{a, b, e, \langle, \rangle\}, M, T_0, \$)$,

где M представляется управляющей табл. 2.7.

Табл. 2.7

Маг. сим-ы	Аванцепочки						
	aa	ab	ba	bb	a	b	ε
T_0	$aa'T_1aaa'a'$	$aa'T_1aaa'a'$		$b\langle T_2ba\rangle a'$			
T_1	e		bb'				
T_2			e	bb'			
a	pop	pop			pop		
b			pop	pop		pop	
a'	pass	pass	pass	pass	pass	pass	pass
b'	pass	pass	pass	pass	pass	pass	pass
e	pass	pass	pass	pass	pass	pass	pass
$\langle$	pass	pass	pass	pass	pass	pass	pass
$\rangle$	pass	pass	pass	pass	pass	pass	pass
\$							accept

Для иллюстрации работы только что построенного 2-предсказывающего алгоритма трансляции рассмотрим обработку входной цепочки bba:

$(bba, T_0\$, \varepsilon) \vdash (bba, b\langle T_2ba\rangle a'\$, \varepsilon) \vdash (ba, \langle T_2ba\rangle a'\$, \varepsilon) \vdash (ba, T_2ba\rangle a'\$, \langle) \vdash$

$\vdash (ba, eba\rangle a'\$, \langle) \vdash (ba, ba\rangle a'\$, \langle e) \vdash (a, a\rangle a'\$, \langle e) \vdash (\varepsilon, \rangle\$, \langle e) \vdash$

$\vdash (\varepsilon, a'\$, \langle e\rangle) \vdash (\varepsilon, \$, \langle e\rangle a)$.

Итак, $\Im(bba) = <e>a$. Нетрудно проверить, что $(S, S) \overset{*}{\underset{T}{\Rightarrow}} (bba, <e>a)$.

Замечание 2.2. Если входная грамматика схемы — $LL(1)$-грамматика или сильная $LL(k)$-грамматика, то очевидно, что можно обойтись без построения $LL(k)$-таблиц, как показано на следующем примере.

Пример 2.14. Пусть $T = (\{E, E', T, T', F\}, \{a, +, *, (,)\}, \{a, +, *\}, R, E)$, где

$R = \{$ (1) $E \to TE', TE'$; (5) $T' \to *FT', F*T'$;

(2) $E' \to +TE', T+E'$; (6) $T' \to \varepsilon, \varepsilon$;

(3) $E' \to \varepsilon, \varepsilon$; (7) $F \to (E), E$;

(4) $T \to FT', FT'$; (8) $F \to a, a\}$.

Очевидно, что T — простая, семантически однозначная sdts с входной грамматикой $LL(1)$. Посредством алгоритма 2.7 получаем следующий 1-предсказы-вающий алгоритм трансляции:

$$\Im = (\{a, +, *, (,)\}, \{E, E', T, T', F, a, +, *, (,), a', +', *', \$\}, \{a, +, *\}, M, E, \$),$$

где M представлена в табл. 2.8.

Табл. 2.8

Маг. сим-ы	Аванцепочки					
	a	+	*	(	)	ε
E	TE'			TE'		
E'		$+T+'E'$			ε	ε
T	FT'			FT'		
T'		ε	$*F*'T'$		ε	ε
F	aa'			(E)		
a	pop					
+		pop				
*			pop			
(				pop		
)					pop	
a'	pass					
$+'$						
$*'$						
$						accept

Посмотрим, как будет действовать построенный нами 1-предсказывающий алгоритм трансляции на входной цепочке $(a+a)$[16].

$((a+a), E\$, \varepsilon) \vdash ((a+a), TE'\$, \varepsilon) \vdash ((a+a), FT'E'\$, \varepsilon) \vdash$

$\vdash ((a+a), (E)T'E'\$, \varepsilon) \vdash (a+a), E)T'E'\$, \varepsilon) \vdash (a+a), TE')T'E'\$, \varepsilon) \vdash$

$\vdash (a+a), FT'E')T'E'\$, \varepsilon) \vdash (a+a), aa'T'E')T'E'\$, \varepsilon) \vdash$

$\vdash (+a), a'T'E')T'E'\$, \varepsilon) \vdash (+a), T'E')T'E'\$, a) \vdash (+a), E')T'E'\$, a) \vdash$

$\vdash (+a), +T+'E')T'E'\$, a) \vdash (a), T+'E')T'E'\$, a) \vdash$

$\vdash (a), FT'+'E')T'E'\$, a) \vdash (a), aa'T'+'E')T'E'\$, a) \vdash$

$\vdash (), a'T'+'E')T'E'\$, a) \vdash (), T'+'E')T'E'\$, aa) \vdash$

$\vdash (), +'E')T'E'\$, aa) \vdash (), E')T'E'\$, aa+) \vdash (\varepsilon, T'E'\$, aa+) \vdash$

$\vdash (\varepsilon, E'\$, aa+) \vdash (\varepsilon, \$, aa+).$

Итак, $\Im((a+a)) = aa+$. Нетрудно проверить, что $(E, E) \overset{*}{\underset{T}{\Rightarrow}} ((a+a), aa+)$.

[16] Не следует путать скобки, ограничивающие компоненты конфигурации, со скобками — входными символами.

Теорема 2.10. *Пусть* $T = (N, \Sigma, \Delta, R, S)$ *— простая семантически однозначная схема синтаксически управляемой трансляции с входной грамматикой* G_i *класса* $LL(k)$ *и* $\Im = (\Sigma, \Gamma \cup \{\$\}, \Delta, M, T_0, \$)$ *— k-предсказывающий алгоритм трансляции, построенный посредством алгоритма* 2.7, *примененного к данной схеме трансляции T. Тогда* $\tau(T) = \tau(\Im)$.

Доказательство. Справедливость данного утверждения следует из того факта, что $LL(k)$-анализатор, построенный по входной грамматике данной схемы, является правильным. Модификация, превращающая его в $LL(k)$-транслятор, фактически состоит в том, что когда анализатор, моделируя шаг левостороннего вывода, замещает некоторую $LL(k)$-таблицу $T_{A,L}$ образом правой части правила $A \to x_0 A_1 x_1 A_2 \ldots A_m x_m$ вида $x_0 T_{A_1,L_1} x_1 \; T_{A_2,L_2} \ldots T_{A_m,L_m} x_m$, транслятор "подмешивает" к этой магазинной цепочке выходные символы из семантической цепочки соответствующего правила схемы таким образом, что полученная смесь $x_0 y_0' T_{A_1,L_1} x_1 y_1' T_{A_2,L_2} \ldots T_{A_m,L_m} x_m y_m'$ в магазине обеспечивает точное воспроизведение движений анализатора и синхронизированную с ними генерацию соответствующей цепочки на выходной ленте. Действительно, выполнив pop-движения и продвинувшись по фрагменту x_i входной цепочки, который является также и фрагментом правила входной грамматики $A \to x_0 A_1 x_1 A_2 \ldots A_m x_m$, транслятор тут же выдает на выход соответствующий фрагмент y_i выходной цепочки, который также является и фрагментом семантической цепочки соответствующего правила схемы $A \to x_0 A_1 x_1 A_2 \ldots A_m x_m, \; y_0 A_1 y_1 A_2 \ldots A_m y_m$.

Теорема 2.11. *Пусть* $T = (N, \Sigma, \Delta, R, S)$ *— простая семантически однозначная схема синтаксически управляемой трансляции с входной грамматикой* G_i *класса* $LL(k)$. *Существует детерминированный магазинный преобразователь P, такой, что* $\tau_e = \{(x\$, y) \mid (x, y) \in \tau(T)\}$.

Доказательство. Заметим, что входная цепочка трансляции, реализуемой магазинным преобразователем, всегда имеет маркер конца, тогда как схема порождает трансляцию с входами без такого маркера. Маркер необходим, чтобы гарантировать детерминизм магазинного преобразователя.

Доказательство основано на построении dpdt, моделирующего движения k-предсказывающего алгоритма трансляции, адекватного данной схеме, который согласно теореме 2.6 существует.

Итак, пусть построен k-предсказывающий алгоритм трансляции

$$\Im = (\Sigma, \Delta, \Gamma \cup \{\$\}, M, T_0, \$).$$

Положим dpdt

$$P = (Q, \Sigma \cup \{\$\}, \Gamma \cup \{\$\}, \Delta, \delta, q_0, \$, \varnothing),$$

где Σ и Γ — те же, что и в $\Im$; $Q = \{q_0\} \cup \{[u] \mid u \in \Sigma^{*k}\} \cup \{[v\$] \mid v \in \Sigma^{*k-1}\}$.

1. $\delta([\varepsilon], T_0\$, \varepsilon)$ — моделирует начальную конфигурацию $\Im$.

2. $\delta([v], a, T) = ([va], T, \varepsilon)$, $v \in \Sigma^{*k-1}$, $a \in \Sigma$, $T \in \mathcal{T}$ — накопление аванцепочки.

3. $\delta([v], \$, T) = ([v\$], T, \varepsilon)$, $v \in \Sigma^{*k-1}$, $T \in \mathcal{T}$ — накопление короткой аванцепочки.

4. $\delta([u], \varepsilon, T) = ([u], \beta, \varepsilon)$, $u \in \Sigma^{*k}$, $T \in \mathcal{T}$, $M(T, u) = \beta$ — моделирование движения типа 1 .

5. $\delta([v\$], \varepsilon, T) = ([v\$], \beta, \varepsilon)$, $v \in \Sigma^{*k-1}$, $T \in \mathcal{T}$, $M(T, v) = \beta$ — моделирование движения типа 1 при короткой аванцепочке.

6. $\delta([av], \varepsilon, a) = ([v], \varepsilon, \varepsilon)$, $a \in \Sigma$, $v \in \Sigma^{*k-1}$ — моделирование pop-движения.

7. $\delta([av\$], \varepsilon, a) = ([v\$], \varepsilon, \varepsilon)$, $a \in \Sigma$, $v \in \Sigma^{*k-2}$ — моделирование pop-движения при короткой аванцепочке.

8. $\delta([u], \varepsilon, b') = ([u], \varepsilon, b)$, $b \in \Delta$, $v \in \Sigma^{*k}$ — моделирование pass-движения.

9. $\delta([v\$], \varepsilon, b') = ([v\$], \varepsilon, b)$, $b \in \Delta$, $u \in \Sigma^{*k-1}$ — моделирование pass-движения при короткой аванцепочке.

10. $\delta([\$], \varepsilon, \$) = ([\varepsilon], \varepsilon, \varepsilon)$ — моделирование перехода в принимающую конфигурацию.

То, что построенный dpdt P действительно точно моделирует k-предсказывающий алгоритм трансляции $\Im$, нетрудно доказать индукцией по числу движений типа 1 того и другого устройств.

2.11. Непростые *LL*(*k*)-трансляции и магазинные процессоры

Пусть $T = (N, \Sigma, \Delta, R, S)$ — непростая семантически однозначная sdts с входной грамматикой G_i класса $LL(k)$. Для реализации такой трансляции можно ввести еще одну модификацию k-предсказывающего алгоритма анализа, называемую *магазинным процессором*.

Магазинный процессор ведет анализ входной цепочки во входной грамматике и генерирует дерево вывода выходной цепочки в выходной грамматике. Другими словами, он моделирует вывод элемента трансляции, используя магазин для манипуляции над синтаксическими цепочками трансляционных форм, а семантические цепочки этих форм представляет в виде дерева вывода в выходной грамматике, разрастающегося в ходе вывода. В тот момент, когда в процессе анализа он воспроизводит шаг замены крайнего левого нетерминала в синтаксической цепочке текущей трансляционной формы, происходит пристраивание вершин, помеченных символами соответствующей семантической цепочки используемого правила схемы к вершине дерева вывода, представляющей связанное вхождение одноименного нетерминала в семантической цепочке трансляционной формы. Чтобы следить за связями между нетерминалами синтаксической цепочки текущей трансляционной формы с соответствующими вершинами дерева, представляющего семантические цепочки, используются указатели,

хранимые в магазине анализатора при нетерминалах или $LL(k)$-таблицах, ассоциированных с ними. Когда нетерминал (или $LL(k)$-таблица) на вершине магазина подменяется цепочкой, представляющей синтаксический элемент правила схемы, указатель, находящийся при нем (ней), указывает на вершину, к которой надлежит пристроить новые вершины. Указатели же на эти вновь появившиеся вершины помещаются в магазин при связанных вхождениях нетерминалов в синтаксическом элементе правила, о котором шла речь.

Проще всего проиллюстрировать работу магазинного процессора схематически — см. рис. 2.3).

На рис. 2.3, *а* представлена начальная конфигурация магазинного процессора. В магазине кроме маркера "дна" магазина находится только начальная $LL(k)$-таблица T_0 с указателем p_0 на корень строящегося дерева вывода в выходной грамматике результата трансляции входной цепочки. Этот указатель запоминается также в отдельной переменной (Root), чтобы через него получить доступ к дереву после его построения. На рис. 2.3, *б* представлена промежуточная конфигурация магазинного процессора, когда часть дерева вывода построена. Среди его листьев находится вершина, помеченная нетерминалом A. В рассматриваемый момент на вершине магазина находится $LL(k)$-таблица $T_{A,L}$ с указателем p на вершину дерева вывода, к которой будет пристроено поддерево, представляющее семантический элемент правила $A \to \alpha_1 B \beta_1,\ \alpha_2 B \beta_2$, используемого на этом шаге моделирования вывода. В правиле явно выделена пара связанных вхождений одного нетерминала (B). Рис. 2.3, *в* иллюстрирует результат

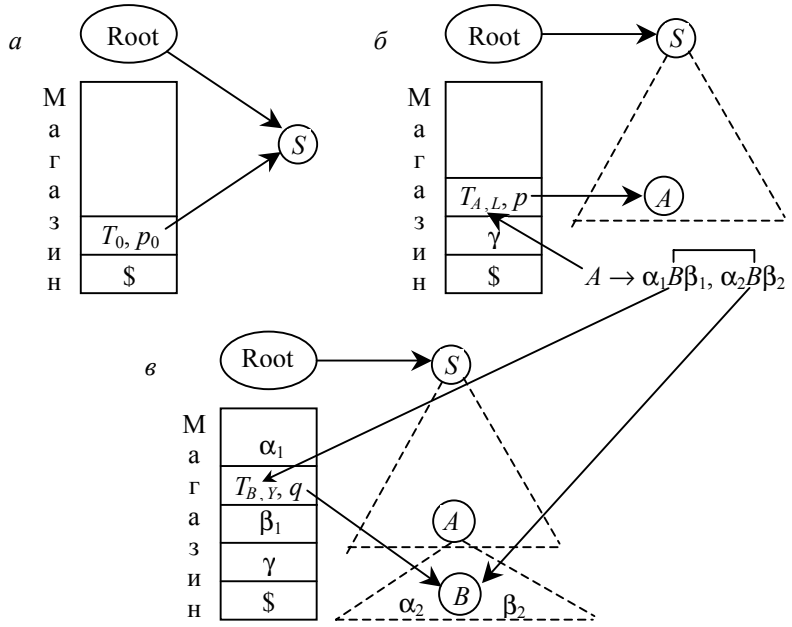

Рис. 2.3.

этого шага: в магазине вместо таблицы $T_{A,L}$ помещен образ синтаксической цепочки $\alpha_1 B \beta_1$, а к узлу A пристроен древовидный образ семантической цепочки $\alpha_2 B \beta_2$. Показана связь одного символа $T_{B,Y}$ — образа нетерминала B в синтаксической цепочке, заместившей в магазине $T_{A,L}$, с узлом B из множества вновь образованных узлов, составляющих образ семантической цепочки $\alpha_2 B \beta_2$. Указатель q определяет ту вершину, к которой будет "подвешиваться" древовидный образ семантической цепочки некоторого правила схемы, когда магазинный символ $T_{B,Y}$ будет замещаться синтаксической цепочкой этого же правила.

Опишем теперь неформально алгоритм построения магазинного процессора по данной схеме, обладающей указанными свойствами.

Алгоритм 2.9: построение магазинного процессора по непростой семантически однозначной sdts с входной грамматикой класса $LL(k)$.

Вход: $T = (N, \Sigma, \Delta, R, S)$ — непростая семантически однозначная sdts с входной грамматикой G_i класса $LL(k)$.

Выход: магазинный процессор $\mathcal{P}$, такой, что $\tau(\mathcal{P}) = \tau(T)$.

Метод.

Магазинный процессор $\mathcal{P}$ в своем магазине будет повторять действия $LL(k)$-анализатора $\mathcal{A}$, построенного по входной грамматике схемы T, используя вместо выходной ленты устройство памяти прямого доступа, в котором он строит дерево вывода результата трансляции в выходной грамматике схемы.

1. Первоначально $\mathcal{P}$ имеет запись (S, p_r)[17] на вершине магазина, где p_r — указатель на корневой узел n_r дерева результата. Этот же указатель дублируются переменной Root.

2. Если $\mathcal{A}$ имеет терминал входной грамматики на вершине магазина и текущий входной символ (первый символ аванцепочки) — такой же терминал, то $\mathcal{A}$ совершает pop-движение и процессор $\mathcal{P}$ делает то же самое.

3. Если $\mathcal{A}$ раскрывает нетерминал A (или замещает $LL(k)$-таблицу, ассоциированную с A) посредством правила $A \to X_1 X_2 \ldots X_m$ с семантическим элементом $y_0 B_1 y_1 \ldots B_m y_m$ и при этом рядом с этим нетерминалом на вершине магазина процессора находится указатель на узел n, то процессор `P` делает следующее:

а) создает узлы прямых потомков для n, помеченных слева направо символами цепочки $y_0 B_1 y_1 \ldots B_m y_m$;

б) на вершине магазина заменяет A (или соответствующую $LL(k)$-таблицу) и указатель при нем на цепочку $X_1 X_2 \ldots X_m$ с указателями при каждом нетерминале, встречающемся в ней; указатель при X_j, если это — нетерминал, указывает на узел, созданный для B_i, если X_j и B_i связаны в правиле схемы $A \to X_1 X_2 \ldots X_m, y_0 B_1 y_1 \ldots B_m y_m$.

---

[17] В случае использования $LL(k)$-таблиц вместо начального нетерминала S будет использоваться начальная $LL(k)$-таблица $T_0 = T_{S,\{\varepsilon\}}$.

4. Если магазин процессора пуст к моменту достижения конца входной цепочки, то он принимает ее. Выход есть дерево с корнем n_r, построенное процессором. Его расположение зафиксировано переменной Root.

Можно показать, что такой алгоритм реализует правильную трансляцию и что на подходящей машине прямого доступа он может быть выполнен за время, линейно зависящее от длины входа.

Теорема 2.12. *Алгоритм* 2.9 *строит магазинный процессор, выдающий в качестве выхода дерево*, *метки листьев которого, выписанные слева направо*, *представляют результат трансляции входной цепочки*.

Глава 3

LR(*k*)-ГРАММАТИКИ И ТРАНСЛЯИИ

§ 3.1. Синтаксический анализ типа “снизу—вверх”

В предыдущей главе был рассмотрен класс *LL*(*k*)-грамматик, наибольший подкласс КС-грамматик, естественным образом детерминированно анализируемых в технике “сверху—вниз”, которая предполагает последовательное построение сентенциальных форм левостороннего вывода, начиная с начальной формы *S*, причем *S* — начальный нетерминал грамматики, и заканчивая конечной формой — анализируемой терминальной цепочкой. Один шаг этого процесса состоит в определении того правила, правая часть которого должна использоваться для замены крайнего левого нетерминала в текущей сентенциальной форме, чтобы получить следующую сентенциальную форму. Именно, если $wA\alpha$ — текущая сентенциальная форма, где w — закрытая часть, $A\alpha$ — открытая часть и x — анализируемая цепочка, то правило определяется по нетерминалу A, множеству допустимых локальных правых контекстов $L = \mathrm{FIRST}_k^G(\alpha)$ и аванцепочке $u = \mathrm{FIRST}_k(y)$, причем $x = wy$. Адекватный тип анализатора — k-предсказывающий алгоритм анализа, выполняя эти шаги, воспроизводит открытые части сентенциальных форм в своем магазине.

В альтернативной терминологии такого рода анализатор выстраивает дерево вывода анализируемой цепочки, начиная с корня и на каждом шаге пристраивая очередное поддерево к крайнему левому нетерминальному листу строящегося дерева (рис. 3.1).

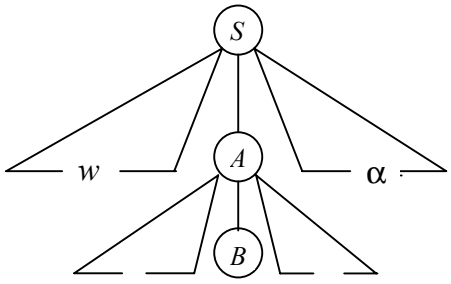

Рис. 3.1.

В противоположность этому подходу техника анализа “снизу—вверх” основана на воспроизведении сентенциальных форм правостороннего вывода, начиная с последней — анализируемой цепочки и заканчивая первой — начальным нетерминалом грамматики. Именно: пусть

$$S = \alpha_0 \underset{\text{rm}}{\overset{p_1}{\Longrightarrow}} \alpha_1 \underset{\text{rm}}{\overset{p_2}{\Longrightarrow}} \ldots \underset{\text{rm}}{\overset{p_n}{\Longrightarrow}} \alpha_n = x$$

— правосторонний вывод терминальной цепочки x в некоторой КС-граммати-ке. Индекс p_i ($i = 1, 2, \ldots, n$) над стрелкой означает, что на данном шаге нетерминал текущей сентенциальной формы α_{i-1} замещается правой частью правила номер p_i. Индекс rm (right-most) под стрелкой означает, что замещается крайний правый нетерминал. Последовательность номеров правил $\pi^R = p_n \ldots \ldots p_2 p_1$ называется *правосторонним анализом цепочки* x.

Задача анализатора типа "снизу—вверх", называемого также *восходящим анализатором*, состоит в том, чтобы найти правосторонний анализ данной входной цепочки x в данной КС-грамматике G. Как было сказано, исходная сентенциальная форма, с которой анализатор начинает работу, есть $\alpha_n = x$. Далее он должен строить последующие сентенциальные формы и заканчивать свою работу тогда, когда будет построена последняя сентенциальная форма $\alpha_0 = S$.

Рассмотрим один шаг работы такого анализатора. Пусть $\alpha_i = \alpha Aw$ — текущая сентенциальная форма правостороннего вывода. Эта форма получается из предыдущей: $\alpha_{i-1} = \gamma Bz \Rightarrow \gamma\beta Ayz = \alpha Aw = \alpha_i$ посредством правила вида $B \to \beta Ay$, где $\alpha = \gamma\beta$, $w = yz$. Как всюду в этом пособии мы обозначаем прописными буквами латинского алфавита нетерминалы, строчными латинскими буквами из конца алфавита — терминальные цепочки, а греческими буквами — цепочки над терминальными и нетерминальными символами.

Размещение сентенциальной формы в восходящем анализаторе показано в табл. 3.1. Восходящий анализатор располагает текущую сентенциальную форму α_i в магазине и на входной ленте таким образом, что в магазине располагается открытая ее часть, а закрытая представлена еще не прочитанной частью входной цепочки, которая начинается с текущего входного символа и простирается до конца этой цепочки.

Табл. 3.1

№	Магазин	Вход
1	$\alpha_i = \alpha A$	w
2	$\gamma\beta A$	yz
3	$\gamma\beta \underline{Ay}$	z
4	$\alpha_{i-1} = \gamma B$	z

В строке 1 табл. 3.1 представлено расположение текущей сентенциальной формы α_i, в строке 2 — то же самое, но более детально. Предполагается, что вершина магазина расположена справа, а текущий входной символ — слева. Анализатор посимвольно сдвигает часть входной цепочки в магазин пока не достигнет правой границы цепочки, составляющей правую часть правила, при помощи которого данная сентенциальная форма α_i была получена из предыдущей α_{i-1}. В строке 3 представлено размещение сентенциальной формы после сдвига в магазин части входа — цепочки y. Далее анализатор сворачивает часть цепочки, примыкающую к вершине магазина и совпадающую с правой частью упомянутого правила, в нетерминал левой части этого правила. В строке 4 при-

веден результат свертки цепочки βAy, располагавшейся на предыдущем шаге в верхней части магазина, в нетерминальный символ B посредством правила $B \to \beta Ay$. Цепочка, подлежащая свертке, называется *основой*: в таблице 3.1 она подчеркнута в строке 3.

Итак, один шаг работы анализатора типа "снизу—вверх" состоит в последовательном сдвиге символов из входной цепочки в магазин до тех пор, пока не достигается правая граница основы, а затем у него должна быть возможность однозначно определить, где в магазине находится левая граница основы и по какому правилу (к какому нетерминалу) свернуть ее. Таким образом он воспроизводит предыдущую сентенциальную форму правостороннего вывода анализируемой цепочки. Если задача решается детерминированным анализатором, то свойства КС-грамматики, в которой производится анализ, должны гарантировать упомянутую однозначность,

Процесс заканчивается, когда в магазине остается один символ S, а входная цепочка прочитана вся.

Замечание 3.1. Если $S \underset{rm}{\overset{*}{\Rightarrow}} \alpha Aw$, то основа β не может быть в пределах α. Действительно, если бы $\alpha = \alpha_1\beta\alpha_2$, то предыдущая сентенциальная форма имела бы вид $\alpha_1 B\alpha_2 Aw$ и из нее текущая получалась бы заменой нетерминала B на β. Но символ B не является крайним правым нетерминалом, что противоречило бы предположению о том, что мы имеем дело с правосторонним выводом. Однако основа могла бы быть в пределах цепочки w или даже цепочки Aw.

Пример 3.1. Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика, в которой $V_N = \{S\}$, $V_T = \{a, b\}$, $P = \{(1)\ S \to SaSb, (2)\ S \to \varepsilon\}$.

Рассмотрим правосторонний вывод в этой грамматике:

$$S \underset{rm}{\overset{(1)}{\Rightarrow}} SaSb \underset{rm}{\overset{(1)}{\Rightarrow}} SaSaSbb \underset{rm}{\overset{(2)}{\Rightarrow}} SaSabb \underset{rm}{\overset{(2)}{\Rightarrow}} Saabb \underset{rm}{\overset{(2)}{\Rightarrow}} aabb.$$

Здесь $\pi^R = 22211$ — правосторонний анализ цепочки $x = aabb$.

Последовательность конфигураций восходящего анализатора во время разбора этой цепочки показана в табл. 3.2.

Табл. 3.2

№	Магазин	Вход	Действие
1	$	*aabb*	reduce 2
2	$*S*	*aabb*	shift
3	$*Sa*	*abb*	reduce 2
4	$*SaS*	*abb*	shift
5	$*SaSa*	*bb*	reduce 2
6	$*SaSaS*	*bb*	shift
7	$*SaSaSb*	*b*	reduce 1
8	$*SaS*	*b*	shift
9	$*SaSb*		reduce 1
10	$*S*		

"Дно" магазина отмечено маркером $. Исходная конфигурация характеризуется тем, что магазин пуст (маркер "дна" не считается), непросмотренная часть входа — вся цепочка *aabb*. Первое действие — свертка пустой основы на вер-

шине магазина по правилу 2. Это приводит к конфигурации, показанной в строке 2. Следующее действие по команде shift — сдвиг: текущий символ *a* перемещается со входа на вершину магазина. Положение вершины магазина тоже изменяется, как и текущий входной символ. Эта конфигурация представлена в строке 3. Дальнейшие действия "сдвиг—свертка" в конце концов приводят к заключительной конфигурации: в магазине — начальный нетерминал грамматики, и вся входная цепочка просмотрена. Номера правил при командах свертки (reduce) образуют правосторонний анализ входной цепочки *aabb*.

Не все КС-грамматики поддаются правостороннему анализу посредством детерминированного механизма типа "перенос—свертка".

Мы рассмотрим здесь класс КС-грамматик, которые позволяют однозначно разрешать упомянутые проблемы путем заглядывания на k символов, следующих за основой. Именно: (1) производить сдвиг или свертку? (2) если делать свертку, то по какому правилу? (3) когда закончить процесс?. Этот класс грамматик, открытый Д.Кнутом, называется *LR*(k)-*грамматиками*. В этом названии *L* обозначает направление просмотра входной цепочки слева направо, *R* — результатом является правосторонний анализ, k — предельная длина аванцепочки.

§ 3.2. *LR*(k)-Грамматики

В этом параграфе мы дадим строгое определение *LR(k)*-грамматик и опишем характерные их свойства.

***Определение* 3.1.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. *Пополненной грамматикой*, полученной из G, назовем грамматику $G' = (V_N', V_T', P', S')$, где $V_N' = V_N \cup \{S'\}$; $S' \notin V_N$; $V_T' = V_T$; $P' = P \cup \{S' \to S\}$.

***Определение* 3.2.** Пусть $G' = (V_N', V_T', P', S')$ — пополненная грамматика для контекстно-свободной грамматики G. Грамматика G является *LR*(k)-*грамматикой*, если для любых двух правосторонних выводов вида

1) $S' \overset{*}{\underset{rm}{\Longrightarrow}} \alpha A w \underset{rm}{\Longrightarrow} \alpha\beta w$,

2) $S' \overset{*}{\underset{rm}{\Longrightarrow}} \gamma B x \underset{rm}{\Longrightarrow} \alpha\beta y$,

в которых $\mathrm{FIRST}_k(w) = \mathrm{FIRST}_k(y)$, должно быть $\alpha A y = \gamma B x$. Другими словами, $\alpha = \gamma$, $A = B$, $y = x$.

Говоря неформально, если согласно первому выводу β — основа, сворачиваемая в нетерминал A, то и во втором выводе β должна быть основой, сворачиваемой в нетерминал A.

Из этого определения следует, что если имеется правовыводимая цепочка $\alpha_i = \alpha\beta w$, где β — основа, полученная из A, и если $\alpha\beta = X_1 X_2 \ldots X_r$, то

1) зная первые символы $X_1 X_2 \ldots X_j$ цепочки $\alpha\beta$ и не более, чем k следующих символов цепочки $X_{j+1} X_{j+2} \ldots X_r w$, мы можем быть уверены, что правый конец основы не будет достигнут до тех пор, пока $j \neq r$;

2) зная цепочку $\alpha\beta = X_1X_2\ldots X_r$ и не более k символов цепочки w, мы можем быть уверены, что именно β является основой, сворачиваемой в нетерминал A;

3) если $\alpha_{i-1} = S'$, можно сигнализировать о выводимости исходной терминальной цепочки из S' и, следовательно, из S.

Использование в определении $LR(k)$-грамматики пополненной грамматики существенно для однозначного определения конца анализа. Действительно, если грамматика использует S в правых частях правил, то свертка основы в S не может служить сигналом приема входной цепочки. Свертка же в S' в пополненной грамматике служит таким сигналом, поскольку S' нигде, кроме начальной сентенциальной формы, не встречается.

Существенность использования пополненной грамматики в определении $LR(k)$-грамматик продемонстрируем на следующем конкретном примере.

Пример 3.2. Пусть пополненная грамматика имеет следующие правила:

0) $S' \to S$,

1) $S \to Sa$,

3) $S \to a$.

Если игнорировать 0-е правило, то, не заглядывая в правый контекст основы Sa, можно сказать, что она должна сворачиваться в S. Аналогично основа a безусловно должна сворачиваться в S. Создается впечатление, что данная грамматика без 0-го правила есть $LR(0)$-грамматика.

С учетом же 0-го правила имеем:

1) $S' \underset{\text{rm}}{\Longrightarrow} S$,

2) $S' \underset{\text{rm}}{\Longrightarrow} S \underset{\text{rm}}{\Longrightarrow} Sa$.

Сопоставив эти два вывода с образцом определения (образец приведен в рамке),

> Если существуют правосторонние выводы
> 1) $S' \overset{*}{\underset{\text{rm}}{\Longrightarrow}} \alpha Aw \underset{\text{rm}}{\Longrightarrow} \alpha\beta w$,
> 2) $S' \overset{*}{\underset{\text{rm}}{\Longrightarrow}} \gamma Bx \underset{\text{rm}}{\Longrightarrow} \alpha\beta y$
> } в которых $\text{FIRST}_k(w) = \text{FIRST}_k(y)$,
> то должно быть $\alpha Ay = \gamma Bx$.

получаем $\alpha = \varepsilon, \beta = S, w = \varepsilon, A = S', \gamma = \varepsilon, B = S, x = \varepsilon, y = a$, и при $k = 0$ $\text{FIRST}_0(w) = \text{FIRST}_0(\varepsilon) = \varepsilon$, $\text{FIRST}_0(y) = \text{FIRST}_0(a) = \varepsilon$, $\alpha Ay = S'a$, $\gamma Bx = S$, а так как $S'a \neq S$, то требование $\alpha Ay = \gamma Bx$ не выполняется.

Итак, данная грамматика не является $LR(0)$-грамматикой.

Лемма 3.1. *Пусть* $G = (V_N, V_T, P, S)$ — $LR(k)$-*грамматика и* G' — пополненная грамматика для G. Если существуют правосторонние выводы в G' вида

1) $S' \overset{*}{\underset{\text{rm}}{\Longrightarrow}} \alpha Aw \underset{\text{rm}}{\Longrightarrow} \alpha\beta w$,

2) $S' \overset{*}{\underset{\text{rm}}{\Longrightarrow}} \gamma Bx \underset{\text{rm}}{\Longrightarrow} \gamma\beta' x = \alpha\beta y$,

в которых $\text{FIRST}_k(w) = \text{FIRST}_k(y)$, *то* $\gamma = \alpha$, $B = A$, $x = y$, $\beta' = \beta$.

Доказательство. Заметим, что первые три равенства непосредственно следуют из определения 3.2 и остается установить только, что $\beta' = \beta$.

Если $\gamma\beta' x = \alpha\beta y$, то в силу первых трех равенств имеем $\gamma\beta' x = \alpha\beta' y = \alpha\beta y$. Следовательно, $\beta' = \beta$. Что и требовалось доказать.

Пример 3.3. Пусть грамматика G содержит следующие правила:

1) $S \to C \mid D$; 2) $C \to aC \mid b$; 3) $D \to aD \mid c$.

Спрашивается, является ли она $LR(0)$-грамматикой.

Отметим прежде всего, что грамматика G — праволинейна. Это значит, что любая сентенциальная форма содержит не более одного нетерминала, причем правый контекст для него всегда пуст. Очевидно также, что любая сентенциальная форма имеет вид $a^i C$, $a^i b$, $a^i D$, $a^i c$, где $i \geq 0$.

Пополненная грамматика содержит еще одно правило:

(0) $S' \to S$.

В роли нетерминала A могут быть при сопоставлении с образцом определения $LR(k)$-грамматик нетерминалы S' либо C, либо D. Отметим, что нетерминал S в роли нетерминала A нас не интересует, поскольку не существует двух разных правосентенциальных форм, в которых участвовал бы нетерминал S. В любом случае в роли цепочек w и x будет выступать пустая цепочка ε из-за праволинейности грамматики G.

Принимая все это во внимание, проверим, отвечает ли данная грамматика определению $LR(0)$-грамматики. В данном конкретном случае $LR(0)$-условие состоит в том, что если существуют два правосторонних вывода вида

1) $S' \underset{rm}{\overset{*}{\Longrightarrow}} \alpha A \underset{rm}{\Longrightarrow} \alpha\beta$,

2) $S' \underset{rm}{\overset{*}{\Longrightarrow}} \gamma B \underset{rm}{\Longrightarrow} \alpha\beta$,

то должно быть $B = A$ и $\gamma = \alpha$. Другими словами, любая сентенциальная форма должна быть выводима единственным способом. Легко убедиться в том, что любая из указанных возможных сентенциальных форм рассматриваемой грамматики обладает этим свойством, и потому $LR(0)$-условие выполняется и, следовательно, G — $LR(0)$-грамматика.

Очевидно, что данная грамматика G не $LL(k)$-грамматика ни при каком k.

Пример 3.4. Рассмотрим грамматику G с правилами:

1) $S \to Ab \mid Bc$; 2) $A \to Aa \mid \varepsilon$; 3) $B \to Ba \mid \varepsilon$.

Эта леволинейная грамматика порождает тот же самый язык, что и грамматика предыдущего примера, и она не является $LR(k)$-грамматикой ни при каком k. Действительно, рассмотрим, например, два таких правосторонних вывода:

1) $S' \underset{rm}{\Longrightarrow} S \underset{rm}{\Longrightarrow} Ab \underset{rm}{\overset{*}{\Longrightarrow}} Aa^i b \underset{rm}{\Longrightarrow} a^i b$,

2) $S' \underset{rm}{\Longrightarrow} S \underset{rm}{\Longrightarrow} Bc \underset{rm}{\overset{*}{\Longrightarrow}} Ba^i c \underset{rm}{\Longrightarrow} a^i c$.

Здесь цепочки $a^i b$ и $a^i c$ являются правыми контекстами для пустой основы, которая в одном случае сворачивается в нетерминал A, а в другом — в нетерми-

нетерминал B. В какой нетерминал сворачивать основу, можно определить лишь по последнему символу (если он — b, то сворачивать в нетерминал A, если он — c, то сворачивать в нетерминал B), который может отстоять от нее на сколько угодно большое расстояние (в зависимости от выбора i). Следовательно, каким бы большим ни было k, всегда найдется такое i, что $\mathrm{FIRST}_k(a^i b) = \mathrm{FIRST}_k(a^i c)$, но $A \neq B$.

***Определение* 3.3.** Грамматики, в которых существует несколько разных правил, отличающихся только нетерминалами в левой части, называются *необратимыми*.

В примере 3.4 мы имели дело с необратимой грамматикой. Причина, по которой данная грамматика не LR, в том, что правый контекст основы, каким бы длинным он ни был, не дает возможности однозначно определить, в какой нетерминал следует ее сворачивать.

Пример 3.5. Рассмотрим грамматику, иллюстрирующую другую причину, по которой она не $LR(1)$: невозможность однозначно определить, что является основой в правовыводимой сентенциальной форме:

$$1)\ S \to AB,\ 2)\ A \to a,\ 3)\ B \to CD,\ 4)\ B \to aE,\ 5)\ C \to ab,\ 6)\ D \to bb,\ 7)\ E \to bba.$$

В этой грамматике рассмотрим два правосторонних вывода:

$$1)\ S' \underset{rm}{\Longrightarrow} S \underset{rm}{\Longrightarrow} AB \underset{rm}{\Longrightarrow} ACD \underset{rm}{\Longrightarrow} ACbb \underset{rm}{\Longrightarrow} \overbrace{A\underline{ab}}^{\alpha\beta}\overbrace{bb}^{w},$$

$$2)\ S' \underset{rm}{\Longrightarrow} S \underset{rm}{\Longrightarrow} AB \underset{rm}{\Longrightarrow} AaE \underset{rm}{\Longrightarrow} \overbrace{Aa}^{\alpha\beta}\overbrace{\underline{bba}}^{y}.$$

Здесь $\alpha\beta = Aab, w = bb, \beta = ab, y = ba, \beta' = bba$. И хотя $\mathrm{FIRST}_1(w) = \mathrm{FIRST}_1(y) = b$, оказывается, что $\beta \neq \beta'$, а это является нарушением условия $LR(1)$ (см. лемму 3.1).

§ 3.3. *LR*(*k*)-Анализатор

Аналогично тому, как для $LL(k)$-грамматик адекватным типом анализаторов является k-предсказывающий алгоритм анализа, поведение которого диктуется $LL(k)$-таблицами, для $LR(k)$-грамматик адекватным механизмом анализа является $LR(k)$-анализатор, управляемый $LR(k)$-таблицами. Эти $LR(k)$-таблицы являются строчками управляющей таблицы $LR(k)$-анализатора. $LR(k)$-Таблица состоит из двух подтаблиц, представляющих следующие функции:

$f: V_T^{*k} \to \{\text{shift, reduce } i, \text{ accept, error}\}$,

$g: V_N \cup V_T \to \mathcal{T} \cup \{\text{error}\}$,

где V_T — входной алфавит анализатора (терминалы грамматики); V_N — нетерминалы грамматики; $\mathcal{T}$ — множество $LR(k)$-таблиц для G, оно строится по пополненной грамматике G' для $LR(k)$-грамматики G.

Алгоритм 3.1: действия $LR(k)$-анализатора.

Вход: $G = (V_N, V_T, P, S)$ — $LR(k)$-грамматика; $\mathcal{T}$ — множество $LR(k)$-таблиц для G; $T_0 \in \mathcal{T}$ — начальная $LR(k)$-таблица; $x \in V_T^*$ — входная цепочка.

Выход: π^R — правосторонний анализ x.

Метод.

$LR(k)$-Анализатор реализует классический механизм "сдвиг", описанный в параграфе §3.1. Его действия будем описывать в терминах конфигураций, понимая под конфигурацией тройку (α, w, y), где $\alpha \in (V_N \cup V_T \cup \mathcal{T})^*$ — магазинная цепочка; $w \in V_T^*$ — непросмотренная часть входной цепочки; y — выходная цепочка, состоящая из номеров правил грамматики G.

Начальная конфигурация есть (T_0, x, ε). Далее алгоритм действует согласно следующему описанию:

1: сдвиг. Пусть текущая конфигурация есть $(\gamma T, w, \pi)$, где $T \in \mathcal{T}$ — некоторая $LR(k)$-таблица, $T = (f, g)$ и пусть $f(u) = \text{shift}$ для $u = \text{FIRST}_k(w)$.

1.1. $w \neq \varepsilon$, скажем, $w = aw'$, где $a \in V_T$, $w' \in V_T^*$.

1.1.1. $g(a) = T'$. Тогда $LR(k)$-анализатор совершает следующее движение:

$(\gamma T, w, \pi) = (\gamma T, aw', \pi) \vdash (\gamma TaT', w', \pi)$,

воспроизводящее сдвиг.

1.1.2. $g(a) = \text{error}$. Анализатор сигнализирует об ошибке и останавливается.

1.2. $w = \varepsilon$. В этом случае $u = \varepsilon$ и $f(u) = f(\varepsilon) = \text{error}$, так как сдвиг из пустой цепочки в магазин невозможен. Анализатор сигнализирует об ошибке и останавливается.

2: свертка. Пусть текущая конфигурация есть $(\gamma T X_1 T_1 X_2 T_2 \ldots X_m T_m, w, \pi)$, где $T, T_1, T_2, \ldots, T_m \in \mathcal{T}$ — некоторые $LR(k)$-таблицы; $T_m = (f_m, g_m)$, $f_m(u) = \text{reduce } i$, $A \to \alpha$ — i-е правило из множества правил P, где $\alpha = X_1 X_2 \ldots X_m$ — основа, $u = \text{FIRST}_k(w)$, и пусть $T = (f, g)$.

2.1. $g(A) = T'$, где $T' \in \mathcal{T}$ — некоторая $LR(k)$-таблица. В этом случае анализатор совершает переход

$$(\gamma T \underbrace{X_1 T_1 X_2 \ldots X_m T_m}_{\text{Свертка в } A}, w, \pi) \vdash (\gamma TA, w, \pi i) \vdash (\gamma TAT', w, \pi i),$$

воспроизводящий свертку.

2.2. $g(A) = \text{error}$. Анализатор сигнализирует об ошибке и останавливается.

3: ошибка. Пусть текущая конфигурация есть $(\gamma T, w, \pi)$, где $T \in \mathcal{T}$ — некоторая $LR(k)$-таблица; $u = \text{FIRST}_k(w)$, $T = (f, g)$ и $f(u) = \text{error}$. Анализатор сигнализирует об ошибке и останавливается.

4: прием. Пусть текущая конфигурация есть $(T_0 S T, \varepsilon, \pi^R)$, $T = (f, g)$, $f(\varepsilon) = \text{accept}$. Анализатор сигнализирует о приеме цепочки x и останавливается. Выходная цепочка π^R представляет правосторонний анализ цепочки x.

Пример 3.6. Обратимся еще раз к грамматике из примера 3.1. Как мы увидим далее, она — $LR(1)$-грамматика. Она имеет следующие правила: 1) $S \to SaSb$, 2) $S \to \varepsilon$. Управляющая таблица $LR(1)$-анализатора, построенная для нее, имеет вид, представленный табл. 3.3, где пустые клетки соответствуют значениям error.

Табл. 3.3

$LR(1)$-таблицы	$f(u)$			$g(X)$		
	a	b	ε	S	a	b
T_0	reduce 2		reduce 2	T_1		
T_1	shift		accept		T_2	
T_2	reduce 2	reduce 2		T_3		
T_3	shift	shift			T_4	T_5
T_4	reduce 2	reduce 2		T_6		
T_5	reduce 1		reduce 1			
T_6	shift	shift			T_4	T_7
T_7	reduce 1	reduce 1				

Рассмотрим действия этого анализатора на входной цепочке *aabb*:

$$(T_0, aabb, \varepsilon) \vdash (T_0ST_1, aabb, 2) \vdash (T_0ST_1aT_2, abb, 2) \vdash (T_0ST_1aT_2ST_3, abb, 22) \vdash$$
$$\vdash (T_0ST_1aT_2ST_3aT_4, bb, 22) \vdash (T_0ST_1aT_2ST_3aT_4ST_6, bb, 222) \vdash$$
$$\vdash (T_0ST_1aT_2ST_3aT_4ST_6bT_7, b, 222) \vdash (T_0ST_1aT_2ST_3, b, 2221) \vdash$$
$$\vdash (T_0ST_1aT_2ST_3bT_5, \varepsilon, 2221) \vdash (T_0ST_1, \varepsilon, 22211).$$

Итак, цепочка *aabb* принимается, и $\pi^R = 22211$ — ее правосторонний анализ.

§ 3.4. Свойства *LR*(*k*)-грамматик

Рассмотрим некоторые следствия из определения $LR(k)$-грамматик, подводящие нас к механизму анализа.

***Определение* 3.4.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика и $S \underset{rm}{\overset{*}{\Longrightarrow}} \alpha Aw \underset{rm}{\Longrightarrow} \alpha\beta w$ — некоторый правосторонний вывод в грамматике G, где $\alpha, \beta \in (V_N \cup V_T)^*$, $w \in V_T^*$.

Активным префиксом сентенциальной формы $\alpha\beta w$ называется любая начальная часть (префикс) цепочки $\alpha\beta$, включая, в частности, ε и всю эту цепочку.

***Определение* 3.5.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика и $A \to \beta_1\beta_2 \in P$. Композицию $[A \to \beta_1.\beta_2, u]$, где $u \in V_T^{*k}$, назовем $LR(k)$-*ситуацией*.

***Определение* 3.6.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика и $S \underset{rm}{\overset{*}{\Longrightarrow}} \alpha Aw \underset{rm}{\Longrightarrow} \alpha\beta w$ — правосторонний вывод в грамматике G, где $\beta = \beta_1\beta_2$, $\beta_1, \beta_2 \in (V_N \cup V_T)^*$. Назовем $[A \to \beta_1.\beta_2, u]$, где $u = \mathrm{FIRST}_k(w)$, $LR(k)$-*ситуацией*, *допустимой для активного префикса* $\alpha\beta_1$.

Пример 3.7. Обратимся еще раз к $LR(0)$-грамматике из примера 3.3, которая содержит следующие правила:

1) $S \to C \mid D$, 2) $C \to aC \mid b$, 3) $D \to aD \mid c$.

Рассмотрим правосторонний вывод $S \underset{rm}{\Longrightarrow} C$. В сентенциальной форме C основой является C. Эта форма имеет два активных префикса: ε и C. Для активного префикса ε допустима $LR(0)$-ситуация $[S \to .C, \varepsilon]$, а для активного префикса C — $LR(0)$-ситуация $[S \to C., \varepsilon]$.

Рассмотрим выводы, дающие активный префикс $aaaa$, и $LR(0)$-ситуации, допустимые для него:

1) $S \underset{rm}{\overset{*}{\Longrightarrow}} aaaC \underset{rm}{\Longrightarrow} \overbrace{aaa}^{\alpha}\,\overbrace{\underbrace{a}_{\beta_1}\,\underbrace{C}_{\beta_2}}^{\beta}$, $\quad \alpha\beta_1 = aaaa$, $[C \to a.C, \varepsilon]$;

2) $S \underset{rm}{\overset{*}{\Longrightarrow}} aaaaC \underset{rm}{\Longrightarrow} \overbrace{aaaa}^{\alpha}\overbrace{\underbrace{\ }_{\beta_1}\,\underbrace{b}_{\beta_2}}^{\beta}$, $\quad \alpha\beta_1 = aaaa$, $[C \to .b, \varepsilon]$;

3) $S \underset{rm}{\overset{*}{\Longrightarrow}} aaaD \underset{rm}{\Longrightarrow} \overbrace{aaa}^{\alpha}\,\overbrace{\underbrace{a}_{\beta_1}\,\underbrace{D}_{\beta_2}}^{\beta}$, $\quad \alpha\beta_1 = aaaa$, $[D \to a.D, \varepsilon]$;

4) $S \underset{rm}{\overset{*}{\Longrightarrow}} aaaaD \underset{rm}{\Longrightarrow} \overbrace{aaaa}^{\alpha}\overbrace{\underbrace{\ }_{\beta_1}\,\underbrace{c}_{\beta_2}}^{\beta}$, $\quad \alpha\beta_1 = aaaa$, $[D \to .c, \varepsilon]$.

Во всех случаях правый контекст основы пуст.

Лемма 3.2. *Пусть* $G = (V_N, V_T, P, S)$ *— не* $LR(k)$*-грамматика. Тогда существуют два правосторонних вывода в пополненной грамматике*:

1) $S' \underset{rm}{\overset{*}{\Longrightarrow}} \alpha Aw \underset{rm}{\Longrightarrow} \alpha\beta w$;

2) $S' \underset{rm}{\overset{*}{\Longrightarrow}} \gamma Bx \underset{rm}{\Longrightarrow} \gamma\delta x = \alpha\beta y$, *такие, что* $x, y, w \in V_T^*$ *и*

а) $\mathrm{FIRST}_k(w) = \mathrm{FIRST}_k(y)$,

б) $\gamma Bx \neq \alpha Ay$,

в) $|\gamma\delta| \geq |\alpha\beta|$.

Доказательство. Если G — не $LR(k)$-грамматика, то условия а) и б) выполняются как прямое следствие из определения $LR(k)$-грамматик. Условие в) неформально означает, что правая граница основы в последней сентенциальной форме второго вывода удалена от ее начала, по крайней мере, не менее, чем удалена основа в последней сентенциальной форме в первом выводе. Это условие не столь очевидно. Простой обмен ролями этих двух выводов ничего не дает, так как этим приемом мы добьемся только выполнения условий а) и в), но не очевидно, что при этом будет выполнено условие б).

Предположим, что выводы 1 и 2 удовлетворяют условиям а) и б), но условие в) не выполнено, т.е. что $|\gamma\delta| < |\alpha\beta|$. Покажем, что тогда найдется другая пара выводов, которые удовлетворяют всем трем условиям.

Поскольку $\gamma\delta x = \alpha\beta y$, то $|\gamma\delta x| = |\alpha\beta y|$, и, учитывая, что $|\gamma\delta| < |\alpha\beta|$, заключаем: $|x| > |y|$, т.е. $x = zy$ при некотором $z \in V_T^+$, $|z| > 0$. Заметим, что цепочка z является префиксом цепочки x, а не ее окончанием, так как именно цепочка y является окончанием всей сентенциальной формы $\alpha\beta y$. Два разных разбиения одной и той же сентенциальной формы $\gamma\delta x = \alpha\beta y$ представлено на рис. 3.2.

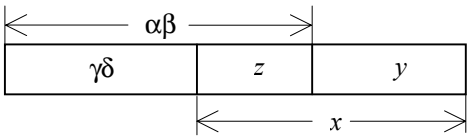

Рис. 3.2.

Условие $\gamma\delta x = \alpha\beta y$ можно переписать как $\gamma\delta zy = \alpha\beta y$, и потому

$$\gamma\delta z = \alpha\beta. \qquad (3.1)$$

Второй вывод разметим по образцу первого с учетом равенства $x = zy$, а первый разметим по образцу второго с учетом равенства (3.1):

$$1') \; S' \underset{rm}{\overset{*}{\Longrightarrow}} \overbrace{\gamma}^{[\alpha]} \overbrace{B}^{[A]} \overbrace{zy}^{[w]} \underset{rm}{\Longrightarrow} \overbrace{\gamma}^{[\alpha]} \overbrace{\delta}^{[\beta]} \overbrace{zy}^{[w]},$$

$$2') \; S' \underset{rm}{\overset{*}{\Longrightarrow}} \overbrace{\alpha}^{[\gamma]} \overbrace{A}^{[B]} \overbrace{w}^{[x]} \underset{rm}{\Longrightarrow} \alpha\beta w = \overbrace{\gamma\delta}^{[\alpha\beta]} \overbrace{zw}^{[y]}.$$

Эти два вывода обладают следующими особенностями:

а′) $\mathrm{FIRST}_k([w]) = \mathrm{FIRST}_k([y])$, так как $[w] = zy$, $[y] = zw$ и $\mathrm{FIRST}_k(w) = \mathrm{FIRST}_k(y)$;

б′) $[\gamma][B][x] \neq [\alpha][A][y]$, так как $[\gamma] = \alpha$, $[B] = A$, $[x] = w$, $[\alpha] = \gamma$, $[A] = B$, $[y] = zw$, поскольку $[\gamma][B][x] = \alpha Aw$ и $[\alpha][A][y] = \gamma Bzw$, а цепочки αAw и γBzw не могут быть равными, так как их терминальные окончания w и zw не равны между собой, ибо $|z| > 0$.

Наконец, выполняется условие

в′) $|[\gamma\delta]| > |[\alpha\beta]|$, ибо $[\gamma\delta] = \alpha\beta$, $[\alpha\beta] = \gamma\delta$ и $|\gamma\delta| < |\alpha\beta|$ по предположению.

Итак, исходная пара правосторонних выводов 1 и 2, которые обменялись ролями, представлены в требуемом виде 1′ и 2′, которые удовлетворяют всем трем условиям а′), б′) и в′). Что и требовалось доказать.

Введем функцию $\mathrm{EFF}_k^G(\alpha)$, где $\alpha \in (V_N \cup V_T)^*$, необходимую для построения $LR(k)$-анализатора. Она будет помогать при определении, является ли ε-це-почка основой для данной сентенциальной формы, подлежащей свертке.

***Определение* 3.7.** Пусть $G = (V_N, V_T, P, S)$ — cfg и $\alpha \in (V_N \cup V_T)^*$. Положим

$$\mathrm{EFF}_k^G(\alpha) = \begin{cases} \mathrm{FIRST}_k^G(\alpha), \text{ если } \alpha \text{ начинается на терминал, а иначе} \\ \{w \in V_T^{*k} \mid \exists \alpha \underset{rm}{\overset{*}{\Longrightarrow}} \beta \underset{rm}{\Longrightarrow} wx, \text{где } \beta \neq Awx, A \in V_N, |w| = k \text{ или } |w| < k \text{ и } x = \varepsilon\}. \end{cases}$$

Говоря неформально, функция $\mathrm{EFF}_k^G(\alpha)$ ничем не отличается от функции $\mathrm{FIRST}_k^G(\alpha)$, если α начинается с терминала, а если α начинается с нетерминала, то при правостороннем выводе именно он является последним нетерминалом, который замещается терминальной цепочкой на последнем шаге вывода. Функция EFF_k^G отличается от функции FIRST_k^G тем, что в первую не включаются префиксы терминальных цепочек, если эти цепочки получены посредством ε-правила, примененного на последнем шаге вывода.

Пример 3.8. Рассмотрим контекстно-свободную грамматику со следующими правилами:

1) $S \to AB$, 2) $A \to Ba \mid \varepsilon$, 3) $B \to Cb \mid C$, 4) $C \to c \mid \varepsilon$.

Вычислим функцию $\mathrm{EFF}_2^G(S)$. Поскольку аргумент начинается на нетерминал, то согласно определению 3.7 необходимо построить всевозможные правосторонние выводы, начинающиеся с нетерминала S и дающие терминальные цепочки, в которых на последнем шаге не применяется ε-правило. В искомое множество нужно включить префиксы этих терминальных цепочек длиной 2 символа, а если они короче, то включить их целиком.

Любой вывод имеет единственное начало: $S \underset{\text{rm}}{\Rightarrow} AB$. Любое продолжение даст результат вида $S \underset{\text{rm}}{\Rightarrow} AB \overset{*}{\underset{\text{rm}}{\Rightarrow}} Aw$, где $w \in V_\text{T}^*$ — некоторая терминальная цепочка. Далее возможны следующие продолжения:

$$S \underset{\text{rm}}{\Rightarrow} AB \overset{*}{\underset{\text{rm}}{\Rightarrow}} Aw \underset{\text{rm}}{\Rightarrow} \begin{cases} \underset{\text{rm}}{\Rightarrow} Cbaw \begin{cases} \underset{\text{rm}}{\Rightarrow} cbaw, \\ \underset{\text{rm}}{\Rightarrow} baw \text{ — недопустимо (ε-правило);} \end{cases} \\ \underset{\text{rm}}{\Rightarrow} Caw \begin{cases} \underset{\text{rm}}{\Rightarrow} caw, \\ \underset{\text{rm}}{\Rightarrow} aw \text{ — недопустимо (ε-правило);} \end{cases} \\ w \text{ — недопустимо (ε-правило).} \end{cases}$$

Таким образом, функция $\mathrm{EFF}_2^G(S) = \{cb,\ ca\}$, тогда как функция $\mathrm{FIRST}_2^G(S) = \{\varepsilon, a, b, c, ab, ac, ba, ca, cb\}$.

Теорема 3.1. *Чтобы контекстно-свободная грамматика* $G = (V_\text{N}, V_\text{T}, P, S)$ *была* $LR(k)$*-грамматикой, необходимо и достаточно, чтобы выполнялось следующее условие: если* $[A \to \beta., u]$ *—* $LR(k)$*-ситуация, допустимая для активного префикса* $\alpha\beta$ *правосентенциальной формы* $\alpha\beta w$ *пополненной грамматики* G'*, где* $u = \mathrm{FIRST}_k(w)$*, то не существует никакой другой* $LR(k)$*-ситуации* $[A_1 \to \beta_1.\beta_2, v]$*, которая отличается от данной и допустима для активного префикса* $\alpha\beta$ *при условии, что* $u \in \mathrm{EFF}_k^G(\beta_2 v)$ *(в частности, и при* $\beta_2 = \varepsilon$*).*

Доказательство.

Необходимость. Предположим, что G — $LR(k)$-грамматика. Докажем, что тогда условие выполняется.

По определению $[A \to \beta., u]$ — $LR(k)$-ситуация, допустимая для активного префикса $\alpha\beta$ правосентенциальной формы $\alpha\beta w$ пополненной грамматики G', если существует правосторонний вывод вида

$$1)\ S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha A w \underset{rm}{\Rightarrow} \alpha\beta w \text{ и } u = \mathrm{FIRST}_k(w).$$

Рассуждая от противного, предположим, что $[A_1 \to \beta_1.\beta_2, v]$ — *другая* $LR(k)$-ситуация, допустимая для активного префикса $\alpha\beta$, причем $u \in \mathrm{EFF}_k^G(\beta_2 v)$.

Согласно определению $LR(k)$-ситуации, допустимой для активного префикса $\alpha\beta$, существует вывод вида

$$2)\ S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1 A_1 x \underset{rm}{\Rightarrow} \alpha_1\beta_1\beta_2 x \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1\beta_1 y = \alpha\beta y,$$

в котором применено правило $A_1 \to \beta_1\beta_2$, и $\alpha_1\beta_1 = \alpha\beta$, $\beta_2 x \underset{rm}{\overset{*}{\Rightarrow}} y$, $v = \mathrm{FIRST}_k(x)$.

Кроме того, выполняется условие

$$3)\ u = \mathrm{FIRST}_k(w) \in \mathrm{EFF}_k^G(\beta_2 v).$$

Условие 3) $u \in \mathrm{EFF}_k^G(\beta_2 v)$ означает согласно определению EFF_k^G, что существует вывод вида $\beta_2 v \underset{rm}{\overset{*}{\Rightarrow}} uz$, в котором $|u| = k$ и $z \in V_T^*$ или $|u| < k$ и $z = \varepsilon$, причем если предпоследняя сентенциальная форма в этом выводе начинается с нетерминала, то он заменяется непустой терминальной цепочкой.

Рассмотрим три возможных варианта состава цепочки β_2: 1) $\beta_2 = \varepsilon$; 2) $\beta_2 \in V_T^+$; 3) β_2 содержит нетерминалы.

Вариант 1: $\beta_2 = \varepsilon$. В этом случае из условия 3) следует, что $u \in \mathrm{EFF}_k^G(v)$ и, следовательно, $u = v$. Соответственно вывод 2) фактически имеет вид

$$2')\ S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1 A_1 x \underset{rm}{\Rightarrow} \alpha_1\beta_1 x = \alpha\beta y.$$

Отсюда $\alpha_1\beta_1 = \alpha\beta$ и $x = y$; соответственно $\mathrm{FIRST}_k(y) = \mathrm{FIRST}_k(x) = v = u$.

Так как $\mathrm{FIRST}_k(w) = u$, то имеем два правосторонних вывода: 1) и 2'), в которых $\mathrm{FIRST}_k(w) = \mathrm{FIRST}_k(y)$.

По предположению $[A \to \beta., u] \neq [A_1 \to \beta_1.\beta_2, v]$, причем $u = v$. Следовательно, либо $A \neq A_1$, либо $\beta \neq \beta_1$ (ведь $\beta_2 = \varepsilon$). Но так как G — $LR(k)$-грамматика, то должно выполняться равенство

$$\alpha_1 A_1 x = \alpha A y, \tag{3.4}$$

в котором $x = y$. При $A \neq A_1$ это невозможно. Если же $A = A_1$, то имеем $\alpha_1 \neq \alpha$, поскольку в этом случае $\beta \neq \beta_1$ при том, что $\alpha_1\beta_1 = \alpha\beta$. Итак, условие (3.4) не соблюдается, и G — не $LR(k)$-грамматика вопреки первоначальному предположению. Полученное противоречие доказывает необходимость сформулированного в теореме условия в данном частном случае.

Вариант 2: $\beta_2 = z$, $z \in V_T^+$ (β_2 — непустая терминальная цепочка). В этом случае вывод 2) имеет вид

2″) $S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1 A_1 x \underset{rm}{\Rightarrow} \alpha_1\beta_1\beta_2 x = \alpha_1\beta_1 zx = \alpha\beta y$, так что $\alpha_1\beta_1 = \alpha\beta$, $y = zx$ и, кроме того, предполагается, что

3″) $u \in \mathrm{EFF}_k^G(\beta_2 x) = \mathrm{EFF}_k^G(zx) = \mathrm{EFF}_k^G(y) = \mathrm{FIRST}_k(y)$, т.е. $u = \mathrm{FIRST}_k(y)$.

Итак, в выводах 1) и 2) $\mathrm{FIRST}_k(w) = \mathrm{FIRST}_k(y)$.

Чтобы грамматика G была $LR(k)$-грамматикой, должно быть $\alpha_1 A_1 x = \alpha A y$ или, что то же самое, $\alpha_1 A_1 x = \alpha A z x$, но это невозможно при $z \neq \varepsilon$. Получается, что G — не $LR(k)$-грамматика, а это противоречит исходному предположению. Данное противоречие доказывает неправомерность предположения о существовании двух разных $LR(k)$-ситуаций, о которых шла речь.

Вариант 3: цепочка β_2 не пуста и содержит, по крайней мере, один нетерминал. Пусть B — самый левый из них, т.е. $\beta_2 = u_1 B \delta$, где $u_1 \in V_\mathrm{T}^*$, $B \in V_\mathrm{N}$, $\delta \in (V_\mathrm{N} \cup V_\mathrm{T})^*$. Имеем:

1) $S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha A w \underset{rm}{\Rightarrow} \alpha\beta w$ и $\mathrm{FIRST}_k(w) = u$.

2) $S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1 A_1 x \underset{rm}{\Rightarrow} \alpha_1\beta_1\beta_2 x = \alpha\beta\beta_2 x$, где $\alpha_1\beta_1 = \alpha\beta$, $\mathrm{FIRST}_k(x) = v$.

3) $u \in \mathrm{EFF}_k^G(\beta_2 v)$.

В этом варианте из условия 3) следует, что существует правосторонний вывод $\beta_2 v = u_1 B \delta v \underset{rm}{\overset{*}{\Rightarrow}} u_1 B u_3 v \underset{rm}{\overset{*}{\Rightarrow}} u_1 y_1 B_1 y_2 u_3 v \underset{rm}{\Rightarrow} u_1 y_1 u_2 y_2 u_3 v$, $u = \mathrm{FIRST}_k(u_1 y_1 u_2 y_2 u_3 v)$. Здесь на последнем шаге используется правило $B_1 \to u_2$, где $u_2 \in V_\mathrm{T}^*$ с гарантией, что $u_1 y_1 u_2 \neq \varepsilon$. Действительно, если $u_1 y_1 = \varepsilon$, то в соответствии с определением EFF_k^G должно быть $u_2 \neq \varepsilon$.

Продолжим вывод 2), учитывая, что $\beta_2 = u_1 B \delta \underset{rm}{\overset{*}{\Rightarrow}} u_1 B u_3 \underset{rm}{\Rightarrow} u_1 y_1 u_2 y_2 u_3$:

2′) $S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1 A_1 x \underset{rm}{\Rightarrow} \alpha_1\beta_1\beta_2 x = \alpha\beta u_1 B \delta x \underset{rm}{\overset{*}{\Rightarrow}} \alpha\beta u_1 B u_3 x \underset{rm}{\overset{*}{\Rightarrow}} \alpha\beta u_1 y_1 B_1 y_2 u_3 x \underset{rm}{\Rightarrow}$

$\underset{rm}{\Rightarrow} \alpha\beta u_1 y_1 u_2 y_2 u_3 x = \alpha\beta y$, где $y = u_1 y_1 u_2 y_2 u_3 x$, $v = \mathrm{FIRST}_k(x)$.

Вычислим $\mathrm{FIRST}_k(y)$:

$$\mathrm{FIRST}_k(y) = \mathrm{FIRST}_k(u_1 y_1 u_2 y_2 u_3 x) = \mathrm{FIRST}_k(u_1 y_1 u_2 y_2 u_3 v) = u.$$

Итак, $\mathrm{FIRST}_k(w) = u$, $\mathrm{FIRST}_k(y) = u$, т.е. $\mathrm{FIRST}_k(w) = \mathrm{FIRST}_k(y)$, и остается проверить, выполняется ли $LR(k)$-условие $\alpha\beta u_1 y_1 B_1 y_2 u_3 x = \alpha A u_1 y_1 u_2 y_2 u_3 x$. Для выполнения этого равенства необходимо, по крайней мере, чтобы $y_2 u_3 x = u_1 y_1 u_2 y_2 u_3 x$, т.е. $u_1 y_1 u_2 = \varepsilon$, чего, как мы выяснили, нет ($u_1 y_1 u_2 \neq \varepsilon$).

Таким образом, $LR(k)$-условие нарушено, и G — не $LR(k)$-грамматика вопреки исходному предположению.

Рассмотренные варианты состава цепочки β_2 исчерпывающе доказывают необходимость сформулированного условия.

Достаточность. Рассуждая от противного, предположим, что условие теоремы выполнено, но G — не $LR(k)$-грамматика. Согласно лемме 3.2 существуют два вывода вида

1) $S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha A w \underset{rm}{\Rightarrow} \alpha\beta w$;

2) $S' \underset{rm}{\overset{*}{\Rightarrow}} \gamma B x \underset{rm}{\Rightarrow} \gamma\delta x = \alpha\beta y$, такие, что $x, y, w \in V_T^*$ и

а) $\mathrm{FIRST}_k(w) = \mathrm{FIRST}_k(y)$,

б) $\gamma B x \neq \alpha A y$,

в) $|\gamma\delta| \geq |\alpha\beta|$.

Не ограничивая общности рассуждений, можно считать, что $\alpha\beta$ — одна из самых коротких цепочек, удовлетворяющих описанным условиям.

Представим вывод 2) иначе, выделив в нем явно начальный участок, на котором получается *последняя* цепочка, причем ее открытая часть не длиннее $|\alpha\beta| + 1$:

$2')\ S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1 A_1 y_1 \underset{rm}{\Rightarrow} \alpha_1\beta_1\beta_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}} \alpha\beta y$,

здесь $|\alpha_1 A_1| \leq |\alpha\beta| + 1$ или, что то же самое, $|\alpha_1| \leq |\alpha\beta| \leq |\gamma\delta|$, $A_1 \to \beta_1\beta_2 \in P$. Цепочка $\beta_1\beta_2$ — основа сентенциальной формы $\alpha_1\beta_1\beta_2 y_1$, причем β_1 — ее префикс такой длины, что выполняется равенство $|\alpha_1\beta_1| = |\alpha\beta|$.

Отметим, что активный префикс длиной $|\alpha\beta|$, для которого допустима хотя бы какая-нибудь $LR(k)$-ситуация, не может получиться из сентенциальной формы, открытая часть которой длиннее $|\alpha\beta|+1$. Действительно, если, например, $|\alpha_1 A_1| > |\alpha\beta| + 1$, т.е. $|\alpha_1| > |\alpha\beta|$, то крайний левый символ основы $\beta_1\beta_2$ находился бы в позиции, по меньшей мере, $|\alpha\beta| + 2$, и эта основа не имела бы никакого касательства к префиксу длиной $|\alpha\beta|$. Очевидно, что в выводе $2')$ $\beta_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}} y$ и $\alpha_1\beta_1 = \alpha\beta$. Таким образом, вывод $2')$ можно переписать в следующем виде:

$2'')\ S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1 A_1 y_1 \underset{rm}{\Rightarrow} \alpha_1\beta_1\beta_2 y_1 = \alpha\beta\beta_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}} \alpha\beta y$.

Из факта существования вывода 1) следует, что $LR(k)$-ситуация $[A \to \beta.,\ u]$, где $u = \mathrm{FIRST}_k(w)$, допустима для активного префикса $\alpha\beta$ правосентенциальной формы $\alpha\beta w$.

Аналогично из факта существования вывода $2'')$ следует, что $LR(k)$-ситуа-ция $[A_1 \to \beta_1.\beta_2,\ v]$, где $v = \mathrm{FIRST}_k(y_1)$, допустима для активного префикса $\alpha\beta$ правосентенциальной формы $\alpha\beta\beta_2 y_1$.

Учитывая условие а), вывод $\beta_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}} y$ и равенство $v = \mathrm{FIRST}_k(y_1)$, заключаем, что

$$u = \mathrm{FIRST}_k(w) = \mathrm{FIRST}_k(y) \in \mathrm{FIRST}_k^G(\beta_2 y_1) = \mathrm{FIRST}_k^G(\beta_2) \oplus_k \mathrm{FIRST}_k^G(y_1) =$$
$$= \mathrm{FIRST}_k^G(\beta_2) \oplus_k \{v\} = \mathrm{FIRST}_k^G(\beta_2) \oplus_k \mathrm{FIRST}_k^G(v) = \mathrm{FIRST}_k^G(\beta_2 v).$$

Остается показать, что $u \in \mathrm{EFF}_k^G(\beta_2 v)$. Действительно, если бы $u \notin \mathrm{EFF}_k^G(\beta_2 v)$, то только потому, что цепочка β_2 начилась бы с нетерминала, который на последнем шаге вывода $\beta_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}} y$ замещался бы ε-цепочкой. Сопоставим исходное представление вывода 2) с его же представлением в виде $2'')$:

2) $S' \underset{rm}{\overset{*}{\Rightarrow}} \gamma Bx \underset{rm}{\Rightarrow} \gamma\delta x = \alpha\beta y$

и

$2''$) $S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1 A_1 y_1 \underset{rm}{\Rightarrow} \alpha_1\beta_1\beta_2 y_1 = \alpha\beta\beta_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}} \alpha\beta y$.

Последним замещаемым нетерминалом в этом выводе является B, причем эта-то последняя замена и дает цепочку $\alpha\beta y$. На завершающем участке этого вывода используется $\beta_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}} y$, так что, если $\beta_2 = A_2\alpha_2$ и последнее используемое правило есть $B \to \varepsilon$, то вывод $2''$) можно переписать следующим образом:

$$S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1 A_1 y_1 \underset{rm}{\Rightarrow} \alpha_1\beta_1\beta_2 y_1 = \alpha_1\beta_1 A_2\alpha_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1\beta_1 A_2 y_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}} \alpha_1\beta_1 A_m\alpha_m y_{m-1}\ldots y_2 y_1 \underset{rm}{\overset{*}{\Rightarrow}}$$
$$\underset{rm}{\overset{*}{\Rightarrow}} \alpha_1\beta_1 A_m y_m y_{m-1}\ldots y_2 y_1 = \alpha_1\beta_1 B y_m y_{m-1}\ldots y_2 y_1 \underset{rm}{\Rightarrow} \alpha_1\beta_1 y_m y_{m-1}\ldots y_2 y_1 = \alpha\beta y,$$

где $\alpha_1\beta_1 = \alpha\beta$, $A_m = B$, $y_m y_{m-1}\ldots y_2 y_1 = y$.

Отметим, что $|\alpha_1\beta_1 B| = |\alpha\beta| + 1$, но это противоречит предположению, что $\alpha_1 A_1 y_1$ — последняя цепочка в этом выводе, открытая часть которой имеет длину, не превосходящую величину $|\alpha\beta| + 1$. Следовательно, $u \in \mathrm{EFF}_k^G(\beta_2 v)$.

Мы нашли две $LR(k)$-ситуации, допустимые для одного и того же активного префикса $\alpha\beta$: $[A \to \beta., u]$ и $[A_1 \to \beta_1.\beta_2, v]$ при том, что $u \in \mathrm{EFF}_k^G(\beta_2 v)$. Поскольку с самого начала предполагалось, что не существует двух таких *разных* $LR(k)$-ситуаций, то должно быть $[A \to \beta., u] = [A_1 \to \beta_1.\beta_2, v]$. Из этого равенства следует: $A = A_1$, $\beta = \beta_1$, $\beta_2 = \varepsilon$. Кроме того, поскольку $\alpha_1\beta_1 = \alpha\beta$, а $\beta_1 = \beta$, то $\alpha_1 = \alpha$. С учетом этого вывод $2''$) можем переписать так:

$2''$) $S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha A y_1 \underset{rm}{\Rightarrow} \alpha\beta y_1 = \alpha\beta y$, откуда заключаем, что $y_1 = y$.

Не забывая, что это другой вид того же самого вывода 2), заменим цепочку β на A, и получим цепочку $\alpha\beta y$, которая совпадает с предыдущей сентенциальной формой в этом выводе. Это означает нарушение условия б) $\gamma Bx \neq \alpha A y$. Данное противоречие — следствие неправомерного допущения, что G — не $LR(k)$-грамматика. Следовательно, G — $LR(k)$-грамматика. Достаточность доказана. Теорема также доказана.

***Определение* 3.8.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика и $\gamma \in (V_N \cup V_T)^*$ — некоторый ее активный префикс. Определим множество $V_k^G(\gamma)$ как множество всех $LR(k)$-ситуаций, допустимых для γ:

$$V_k^G(\gamma) = \{[A \to \beta_1.\beta_2,\ u] \mid \exists A \to \beta_1\beta_2 \in P; \exists S' \underset{rm}{\overset{*}{\Rightarrow}} \alpha A w \underset{rm}{\Rightarrow} \alpha\beta_1\beta_2 w,\ \gamma = \alpha\beta_1,$$
$$u = \mathrm{FIRST}_k(w)\}.$$

Множество $\mathcal{S} = \{\mathcal{A} \mid \mathcal{A} = V_k^G(\gamma)$, где γ — активный префикс $G\}$ назовем *системой множеств $LR(k)$-ситуаций* для грамматики G.

Алгоритм 3.2: вычисление множества $V_k^G(\gamma)$.

Вход: $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика, $\gamma = X_1 X_2 \ldots X_m$, $X_i \in V_N \cup V_T$, $i = 1, 2, \ldots, m$; $m \geq 0$.

Выход: множество $V_k^G(\gamma)$.

Метод.

Будем строить последовательность множеств $V_k^G(\varepsilon)$, $V_k^G(X_1)$, $V_k^G(X_1X_2)$,…, $V_k^G(X_1X_2...X_m)$

1. Строится множество $V_k^G(\varepsilon)$.

а) Инициализация: $V_k^G(\varepsilon) = \{[S\rightarrow.\alpha, \varepsilon] \mid \exists\, S \rightarrow \alpha \in P\}$.

б) Замыкание $V_k^G(\varepsilon)$: если $[A \rightarrow .B\alpha, u] \in V_k^G(\varepsilon)$, где $B \in V_N$, и существует $B \rightarrow \beta \in P$, то $[B \rightarrow .\beta, v]$, где $v \in \mathrm{FIRST}_k^G(\alpha u)$, тоже включается в множество $V_k^G(\varepsilon)$.

в) Шаг б) повторяется до тех пор, пока во множестве $V_k^G(\varepsilon)$ не будут просмотрены все имеющиеся в нем $LR(k)$-ситуации. Замыкание завершается за конечное число шагов, так как множества P и V_T^{*k} конечны.

2. Строится следующий элемент последовательности. Пусть множества $V_k^G(X_1X_2...X_i)$, где $0 \le i < m$, уже построены. Покажем, как построить следующее множество $V_k^G(X_1X_2...X_{i+1})$.

а) Инициализация: если $[A \rightarrow \beta_1.X_{i+1}\beta_2, u] \in V_k^G(X_1X_2...X_i)$, то $LR(k)$-ситуация $[A \rightarrow \beta_1X_{i+1}.\beta_2, u]$ включается в множество $V_k^G(X_1X_2...X_{i+1})$.

б) Замыкание $V_k^G(X_1X_2...X_{i+1})$: если $\{[A \rightarrow \beta_1.B\beta_2, u] \in V_k^G(X_1X_2...X_{i+1})$, где $B \in V_N$, и существует $B \rightarrow \beta \in P$, то $[B \rightarrow .\beta, v]$, где $v \in \mathrm{FIRST}_k^G(\beta_2 u)$, тоже включается в множество $V_k^G(X_1X_2...X_{i+1})$.

в) Шаг б) повторяется до тех пор, пока во множестве $V_k^G(X_1X_2...X_{i+1})$ не будут просмотрены все имеющиеся в нем $LR(k)$-ситуации. Замыкание завершается за конечное число шагов, так как множества P и V_T^{*k} конечны.

3. Шаг 2 повторять до тех пор, пока $i < m$.

4. Процесс завершается при $i = m$; $V_k^G(\gamma) = V_k^G(X_1X_2...X_m)$.

Замечание 3.2. Алгоритм 3.2 не требует использования пополненной грамматики.

***Определение* 3.9.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика. На множестве $LR(k)$-ситуаций в этой грамматике определим функцию: $\mathrm{GOTO}(\mathscr{A}, X) = \mathscr{A}'$, где $\mathscr{A} = V_k^G(\gamma)$ — некоторое множество $LR(k)$-ситуаций, допустимых для активного префикса $\gamma \in (V_N \cup V_T)^*$; $X \in V_N \cup V_T$; $\mathscr{A}' = V_k^G(\gamma X)$.

Очевидно, что эта функция строится попутно с построением множеств $V_k^G(\gamma)$ на шаге 2 алгоритма 3.2: если множество $V_k^G(X_1X_2...X_i)$ уже построено, то

$V_k^G(X_1X_2...X_iX_{i+1}) = \mathrm{GOTO}(V_k^G(X_1X_2...X_i), X_{i+1})$.

Остается ввести лишь обозначения: $\gamma = X_1X_2...X_i$, $X = X_{i+1}$, $\mathcal{A} = V_k^G(X_1X_2...X_i)$, $\mathcal{A}' = V_k^G(X_1X_2...X_iX_{i+1})$, чтобы увидеть, как это делается.

Замечание 3.3. Важно отметить, что результат функции GOTO($\mathcal{A}$, X) = $\mathcal{A}'$, где $\mathcal{A} = V_k^G(X_1X_2...X_i)$, зависит не от $X_1X_2...X_i$, а от $V_k^G(X_1X_2...X_i)$.

Пример 3.9. Рассмотрим пополненную грамматику примера 3.1, содержащую правила: 0) $S' \to S$, 1) $S \to SaSb$, 2) $S \to \varepsilon$. Построим множества $V_1^G(\varepsilon)$, $V_1^G(S)$, $V_1^G(Sa)$.

1: построение множества $V_1^G(\varepsilon)$.

а) $V_1^G(\varepsilon) = \{[S' \to .S, \varepsilon]\}$.

б) Множество $V_1^G(\varepsilon)$ пополняется ситуациями $[S \to .SaSb, \varepsilon]$, $[S \to ., \varepsilon]$;

множество $V_1^G(\varepsilon)$ пополняется ситуациями $[S \to .SaSb, a]$, $[S \to ., a]$.

Другие шаги алгоритма 3.2 никаких других элементов в множество $V_1^G(\varepsilon)$ не добавляют. Окончательно получаем

$V_1^G(\varepsilon) = \{[S' \to .S, \varepsilon], [S \to .SaSb, \varepsilon], [S \to ., \varepsilon], [S \to .SaSb, a], [S \to ., a]\}$.

В сокращенных обозначениях то же самое принято записывать следующим образом:

$V_1^G(\varepsilon) = \{[S' \to .S, \varepsilon], [S \to .SaSb, \varepsilon \mid a], [S \to ., \varepsilon \mid a]\}$.

2: построение множества $V_1^G(S)$.

а) $V_1^G(S) = \{[S' \to S., \varepsilon], [S \to S.aSb, \varepsilon \mid a]\}$.

Так как точка ни в одной из этих ситуаций не стоит перед нетерминалом, то шаг б) не выполняется ни разу. Попутно мы вычислили GOTO($V_1^G(\varepsilon)$, S) = $V_1^G(S)$.

3: построение множества $V_1^G(Sa)$.

а) $V_1^G(Sa) = \{[S \to Sa.Sb, \varepsilon \mid a]\}$.

б) Множество $V_1^G(Sa)$ пополняется ситуациями $[S \to .SaSb, b]$ и $[S \to ., b]$;

множество $V_1^G(Sa)$ пополняется ситуациями $[S \to .SaSb, a]$ и $[S \to ., a]$.

Здесь шаг б) замыкания выполнялся дважды.

Итак, $V_1^G(Sa) = \{[S \to Sa.Sb, \varepsilon \mid a], [S \to .SaSb, a \mid b], [S \to ., a \mid b]\}$ и GOTO($V_1^G(S)$, a) = $V_1^G(Sa)$.

Теорема 3.2. Алгоритм 3.2 правильно вычисляет $V_k^G(\gamma)$, где $\gamma \in (V_N \cup V_T)^*$ — активный префикс правосентенциальной формы в грамматике G.

Доказательство. Фактически требуется доказать, что $LR(k)$-ситуация $[A \to \beta_1.\beta_2, u] \in V_k^G(\gamma)$ тогда и только тогда, когда существует правосторонний

вывод вида $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha Aw \underset{rm}{\Rightarrow} \alpha\beta_1\beta_2 w$, в котором $\alpha\beta_1 = \gamma$ (γ — активный префикс правосентенциальной формы $\alpha\beta_1\beta_2 w$), а $\mathrm{FIRST}_k(w) = u$ (цепочка u есть правый контекст основы $\beta_1\beta_2$ в данной сентенциальной форме $\alpha\beta_1\beta_2 w$).

I. Докажем сначала, что если $\gamma = \alpha\beta_1$ — активный префикс сентенциальной формы $\alpha\beta_1\beta_2 w$ в выводе $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha Aw \underset{rm}{\Rightarrow} \alpha\beta_1\beta_2 w$, то $LR(k)$-ситуация $[A \to \beta_1.\beta_2, u]$, где $u = \mathrm{FIRST}_k(w)$, содержится в множестве $V_k^G(\gamma)$.

Индукция по $l = |\gamma|$.

База. Пусть $l = 0$, т.е. $\gamma = \varepsilon$.

Имеем вывод $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha Aw \underset{rm}{\Rightarrow} \alpha\beta_1\beta_2 w$, $\gamma = \alpha\beta_1 = \varepsilon$. Фактически $\alpha = \beta_1 = \varepsilon$, и вывод имеет вид $S \overset{*}{\underset{rm}{\Rightarrow}} Aw \underset{rm}{\Rightarrow} \beta_2 w$, где на последнем шаге применялось правило $A \to \beta_2$. Во всех деталях он мог бы быть только таким:

$$S \underset{rm}{\Rightarrow} A_1\alpha_1 \overset{*}{\underset{rm}{\Rightarrow}} A_1 w_1 \underset{rm}{\Rightarrow} A_2\alpha_2 w_1 \overset{*}{\underset{rm}{\Rightarrow}} A_2 w_2 w_1 \underset{rm}{\Rightarrow} \ldots \underset{rm}{\Rightarrow} A_m\alpha_m w_{m-1}\ldots w_1 \overset{*}{\underset{rm}{\Rightarrow}}$$
$$\overset{*}{\underset{rm}{\Rightarrow}} A_m w_m w_{m-1}\ldots w_1 = Aw \underset{rm}{\Rightarrow} \beta_2 w.$$

Следовательно, $w = w_m w_{m-1}\ldots w_1$, $A = A_m$ и существуют правила $S \to A_1\alpha_1$, $A_1 \to A_2\alpha_2, \ldots$, $A_{m-1} \to A_m\alpha_m = A\alpha_m$, $A \to \beta_2$. Кроме того, $\mathrm{FIRST}_k(w_i) \in \mathrm{FIRST}_k^G(\alpha_i)$, поскольку $\alpha_i \overset{*}{\underset{rm}{\Rightarrow}} w_i$, $i = 1, 2, \ldots, m$.

Согласно алгоритму 3.2

$[S \to .A_1\alpha_1, \varepsilon] \in V_k^G(\varepsilon)$,

$[A_1 \to .A_2\alpha_2, v_1] \in V_k^G(\varepsilon)$ для любой $v_1 \in \mathrm{FIRST}_k^G(\alpha_1\varepsilon)$,

в частности для $v_1 = \mathrm{FIRST}_k(w_1)$;

$[A_2 \to .A_3\alpha_3, v_2] \in V_k^G(\varepsilon)$ для любой $v_2 \in \mathrm{FIRST}_k^G(\alpha_2 v_1)$,

в частности для $v_2 = \mathrm{FIRST}_k(w_2 v_1) = \mathrm{FIRST}_k(w_2 w_1)$;

…

$[A_{m-1} \to A_m\alpha_m, v_{m-1}] \in V_k^G(\varepsilon)$ для любой $v_{m-1} \in \mathrm{FIRST}_k^G(\alpha_{m-1} v_{m-2})$,

в частности для $v_{m-1} = \mathrm{FIRST}_k(w_{m-1} v_{m-2}) = \mathrm{FIRST}_k(w_{m-1}\ldots w_1)$.

Наконец, поскольку $A_m = A$ и имеется правило $A \to \beta_2$,

$[A \to .\beta_2, v_m] \in V_k^G(\varepsilon)$ для любой $v_m \in \mathrm{FIRST}_k^G(\alpha_m v_{m-1})$,

в частности для $v_m = \mathrm{FIRST}_k(w_m v_{m-1}) = \mathrm{FIRST}_k(w_m w_{m-1}\ldots w_1)$.

Принимая во внимание, что $w = w_m w_{m-1}\ldots w_1$ и что $u = \mathrm{FIRST}_k(w)$, заключаем, что $v_m = u$ и $[A \to .\beta_2, u] \in V_k^G(\varepsilon)$. База доказана.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех активных префиксов γ, таких, что $|\gamma| \le n$ ($n \ge 0$).

Индукционный переход. Покажем, что утверждение выполняется для γ, таких, что $|\gamma| \le n + 1$.

Пусть $\gamma = \gamma' X$, где $|\gamma'| = n$, а $X \in V_N \cup V_T$. Поскольку γ — активный префикс, то существует вывод такой сентенциальной формы, где γ участвует в этой роли:

$$S \underset{rm}{\overset{*}{\Longrightarrow}} \alpha A w \underset{rm}{\Longrightarrow} \alpha\beta_1\beta_2 w = \gamma\beta_2 w = \gamma' X \beta_2 w,$$

здесь на последнем шаге применялось правило $A \to \beta_1\beta_2$, и $\alpha\beta_1 = \gamma = \gamma' X$.

Случай 1: $\beta_1 \neq \varepsilon$.

Поскольку $\alpha\beta_1 = \gamma' X$ и $\beta_1 \neq \varepsilon$, то именно β_1 заканчивается символом X, т.е. $\beta_1 = \beta_1' X$ при некоторой $\beta_1' \in (V_N \cup V_T)^*$. В этом случае имеем

$$S \underset{rm}{\overset{*}{\Longrightarrow}} \alpha A w \underset{rm}{\Longrightarrow} \alpha\beta_1\beta_2 w = \alpha\beta_1' X \beta_2 w = \gamma' X \beta_2 w, \text{ где } \gamma' = \alpha\beta_1', |\gamma'| = n.$$

В соответствии с индукционным предположением, поскольку γ' является активным префиксом последней сентенциальной формы и $|\gamma'| = n$, $[A \to \beta_1'.X\beta_2,\ u] \in V_k^G(\gamma')$, где $u = \mathrm{FIRST}_k(w)$.

Шаг 2а алгоритма 3.2 дает $[A \to \beta_1' X.\beta_2,\ u] = [A \to \beta_1.\beta_2,\ u] \in V_k^G(\gamma' X) = V_k^G(\gamma)$, что и требовалось доказать.

Случай 2: $\beta_1 = \varepsilon$.

Имеем $S \underset{rm}{\overset{*}{\Longrightarrow}} \alpha A w \underset{rm}{\Longrightarrow} \alpha\beta_1\beta_2 w = \alpha\beta_2 w$, причем на последнем шаге вывода применено правило $A \to \beta_2 \in P$, а $\gamma = \alpha = \gamma' X$ и $|\gamma| = n + 1$, т.е. $\gamma' \in (V_N \cup V_T)^*$, $|\gamma'| = n$, $X \in V_N \cup V_T$. Здесь $\mathrm{FIRST}_k(w) = u$. Надо показать, что $LR(k)$-ситуация $[A \to .\beta_2,\ u] \in V_k^G(\gamma)$.

Рассмотрим подробнее этот вывод, чтобы показать, как впервые появляется символ X, завершающий цепочку α, и как образуется правосентенциальная форма $\alpha A w$. В общем случае он имеет следующий вид:

$$\begin{aligned}
S &\underset{rm}{\overset{*}{\Longrightarrow}} \alpha_1 A_1 w_1 \\
&\underset{rm}{\Longrightarrow} \alpha_1\alpha_2' X A_2 \delta_2 w_1 && (\exists\, A_1 \to \alpha_2' X A_2 \delta_2 \in P,\ \alpha_1\alpha_2' = \gamma'), \\
&\underset{rm}{\overset{*}{\Longrightarrow}} \gamma' X A_2 w_2 w_1 && (\exists\, \delta_2 \underset{rm}{\overset{*}{\Longrightarrow}} w_2), \\
&\ldots \\
&\underset{rm}{\Longrightarrow} \gamma' X A_{m-1}\delta_{m-1} w_{m-2} \ldots w_1 && (\exists\, A_{m-2} \to A_{m-1}\delta_{m-1} \in P), \\
&\underset{rm}{\overset{*}{\Longrightarrow}} \gamma' X A_{m-1} w_{m-1} w_{m-2} \ldots w_1 && (\exists\, \delta_{m-1} \underset{rm}{\overset{*}{\Longrightarrow}} w_{m-1}), \\
&\underset{rm}{\Longrightarrow} \gamma' X A_m \delta_m w_{m-1} w_{m-2} \ldots w_1 && (\exists\, A_{m-1} \to A_m\delta_m \in P), \\
&\underset{rm}{\overset{*}{\Longrightarrow}} \gamma' X A_m w_m w_{m-1} w_{m-2} \ldots w_1 && (\exists\, \delta_m \underset{rm}{\overset{*}{\Longrightarrow}} w_m), \\
&= \gamma' X A w && (A_m = A,\ w_m w_{m-1} \ldots w_1 = w), \\
&\underset{rm}{\Longrightarrow} \gamma' X \beta_2 w = \gamma\beta_2 w. && (\exists\, A \to \beta_2 \in P,\ \gamma = \gamma' X).
\end{aligned}$$

Отметим, что в сентенциальной форме $\alpha_1\alpha_2' X A_2 \delta_2 w_1$ за префиксом $\gamma = \alpha_1\alpha_2' X$ может следовать только нетерминал, ибо иначе основа β_2 не могла бы появиться в рассматриваемом выводе непосредственно за этим префиксом, и $LR(k)$-ситуация $[A \to .\beta_2, u]$ не была бы допустима для префикса γ.

По определению $[A_1 \to \alpha_2'.XA_2\delta_2,\ v_1]$, где $v_1 = \mathrm{FIRST}_k(w_1)$, есть $LR(k)$-ситуация, допустимая для γ'.

Поскольку $|\gamma'| = n$, то в соответствии с индукционной гипотезой можно утверждать, что $[A_1 \to \alpha_2'.XA_2\delta_2,\ v_1] \in V_k^G(\gamma')$, а тогда согласно шагу 2а алгоритма $LR(k)$-ситуация $[A_1 \to \alpha_2'X.A_2\delta_2, v_1] \in V_k^G(\gamma'X) = V_k^G(\gamma)$.

Поскольку существуют правило $A_2 \to A_3\delta_3$ и вывод $\delta_2 \underset{\text{rm}}{\overset{*}{\Rightarrow}} w_2$, то согласно шагу 2б $[A_2 \to .A_3\delta_3, v_2] \in V_k^G(\gamma)$, где $v_2 = \mathrm{FIRST}_k(w_2v_1) = \mathrm{FIRST}_k(w_2w_1)$.

Рассуждая далее аналогичным образом, приходим к выводу, что

$[A_{m-1} \to .A_m\delta_m, v_{m-1}] \in V_k^G(\gamma)$, где $v_{m-1} = \mathrm{FIRST}_k(w_{m-1}v_{m-2}) = \mathrm{FIRST}_k(w_{m-1}w_1)$.

Наконец, согласно шагу 2б, поскольку $A_m = A$ и существуют правило $A \to \beta_2$ и вывод $\delta_m \underset{\text{rm}}{\overset{*}{\Rightarrow}} w_m$, заключаем, что $LR(k)$-ситуация $[A \to .\beta_2,\ v_m] \in V_k^G(\gamma)$, где $v_m = \mathrm{FIRST}_k(w_mv_{m-1}) = \mathrm{FIRST}_k(w_mw_{m-1}\ldots w_1) = \mathrm{FIRST}_k(w) = u$.

Итак, $[A \to .\beta_2, u] \in V_k^G(\gamma)$. Что и требовалось доказать.

II. Докажем теперь, что если $[A \to \beta_1.\beta_2,\ u] \in V_k^G(\gamma)$, то существует правосторонний вывод вида $S \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha Aw \underset{\text{rm}}{\Rightarrow} \alpha\beta_1\beta_2w$, в котором $\alpha\beta_1 = \gamma$ есть активный префикс правосентенциальной формы $\alpha\beta_1\beta_2w$, а $\mathrm{FIRST}_k(w) = u$ есть правый контекст основы $\beta_1\beta_2$ в данной сентенциальной форме $\alpha\beta_1\beta_2w$.

Индукция по $l = |\gamma|$.

База. Пусть $l = 0$, т.е. $\gamma = \varepsilon$. В этом случае $\alpha\beta_1 = \gamma = \varepsilon$, следовательно, $\alpha = \varepsilon$, $\beta_1 = \varepsilon$, и надо доказать существование вывода вида $S \underset{\text{rm}}{\overset{*}{\Rightarrow}} Aw \underset{\text{rm}}{\Rightarrow} \beta_2w$, в котором на последнем шаге применяется правило $A \to \beta_2$, а $\mathrm{FIRST}_k(w) = u$.

Имеем $[A \to .\beta_2, u] \in V_k^G(\varepsilon)$. Все $LR(k)$-ситуации из множества $V_k^G(\varepsilon)$ согласно алгоритму 3.2 получаются на шаге 1а или 1б. В общем случае история попадания данной $LR(k)$-ситуации в множество $V_k^G(\varepsilon)$ такова:

$[S \to .\alpha_1, \varepsilon] \in V_k^G(\varepsilon)$ благодаря шагу 1а алгоритма и правилу $S \to \alpha_1 \in P$;

$\alpha_1 = A_1\delta_1$, $\exists\, A_1 \to \alpha_2 \in P$, и шаг 1б дает $[A_1 \to .\alpha_2, v_1] \in V_k^G(\varepsilon)$,

где $v_1 = \mathrm{FIRST}_k(\delta_1)$, т.е. $\exists\ \delta_1 \underset{\text{rm}}{\overset{*}{\Rightarrow}} w_1$ (любой вывод терминальной цепочки можно перестроить в правосторонний) и $v_1 = \mathrm{FIRST}_k(w_1)$;

$\alpha_2 = A_2\delta_2$, $\exists\, A_2 \to \alpha_3 \in P$, и шаг 1б дает $[A_2 \to .\alpha_3, v_2] \in V_k^G(\varepsilon)$,

где $v_2 = \mathrm{FIRST}_k(\delta_2v_1)$, т.е. $\exists\ \delta_2 \underset{\text{rm}}{\overset{*}{\Rightarrow}} w_2$, и $v_2 = \mathrm{FIRST}_k(w_2v_1) = \mathrm{FIRST}_k(w_2w_1)$;

…

$\alpha_m = A_m\delta_m$, $\exists\, A_m \to \alpha_{m+1} \in P$, и шаг 1($b$) дает $[A_m \to .\alpha_{m+1}, v_m] \in V_k^G(\varepsilon)$,

где $v_m = \mathrm{FIRST}_k(\delta_m v_{m-1})$; т.е. $\exists\ \delta_m \underset{\text{rm}}{\overset{*}{\Longrightarrow}} w_m$ и $v_m = \mathrm{FIRST}_k(w_m v_{m-1}) =$

$= \mathrm{FIRST}_k(w_m w_{m-1} \dots w_1) = \mathrm{FIRST}_k(w) = u,\ A_m = A,\ \alpha_{m+1} = \beta_2,$

$w_m w_{m-1} \dots w_1 = w.$

Используя существующие правила и упомянутые частичные выводы, построим требуемый вывод:

$$S \underset{\text{rm}}{\Longrightarrow} \alpha_1 = A_1\delta_1 \underset{\text{rm}}{\overset{*}{\Longrightarrow}} A_1 w_1 \underset{\text{rm}}{\Longrightarrow} \alpha_2 w_1 = A_2\delta_2 w_1 \underset{\text{rm}}{\overset{*}{\Longrightarrow}} A_2 w_2 w_1 \underset{\text{rm}}{\Longrightarrow} \dots \underset{\text{rm}}{\Longrightarrow} A_m\delta_m w_{m-1} \dots w_2 w_1 \underset{\text{rm}}{\overset{*}{\Longrightarrow}}$$
$$\underset{\text{rm}}{\overset{*}{\Longrightarrow}} A_m w_m w_{m-1} \dots w_2 w_1 \underset{\text{rm}}{\Longrightarrow} \alpha_{m+1} w_m w_{m-1} \dots w_2 w_1 = \beta_2 w.$$

Итак, существует вывод $S \underset{\text{rm}}{\overset{*}{\Longrightarrow}} Aw \underset{\text{rm}}{\Longrightarrow} \beta_2 w$ и $\mathrm{FIRST}_k(w) = \mathrm{FIRST}_k(w_m \dots w_1) = u$. База доказана.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех γ, длина которых не превосходит n ($n \geq 0$).

Индукционный переход. Покажем, что утверждение выполняется для $\gamma = \gamma' X$, где $|\gamma'| = n$, $X \in V_N \cup V_T$.

Согласно алгоритму 3.2 $LR(k)$-ситуация $[A \to \beta_1.\beta_2,\ u] \in V_k^G(\gamma)$ в общем случае может быть получена следующим образом:

1. Существует $LR(k)$-ситуация $[A_1 \to \alpha_1.X\alpha_2,\ v_1] \in V_k^G(\gamma')$, допустимая для активного префикса γ'.

2. Посредством шага 2а алгоритма построения множества $V_k^G(\gamma)$ получается $LR(k)$-ситуация $[A_1 \to \alpha_1 X.\alpha_2,\ v_1] \in V_k^G(\gamma' X) = V_k^G(\gamma)$.

Случай 1: $[A_1 \to \alpha_1 X.\alpha_2,\ v_1] = [A \to \beta_1.\beta_2,\ u]$, т.е. рассматриваемая $LR(k)$-ситуация получена на этом шаге алгоритма.

Из того, что $[A_1 \to \alpha_1.X\alpha_2,\ v_1] \in V_k^G(\gamma')$, в соответствии с индукционным предположением следует существование вывода вида

$$S \underset{\text{rm}}{\overset{*}{\Longrightarrow}} \alpha_0 A_1 w_1 \underset{\text{rm}}{\Longrightarrow} \underbrace{\alpha_0}_{\gamma}\overbrace{\underbrace{\alpha_1}_{\gamma} X \underbrace{\alpha_2}_{\beta}}^{\beta} w_1 = \underbrace{\gamma' X}_{\gamma}\beta_2 w_1 = \gamma\beta_2 w_1, \text{ причем } \mathrm{FIRST}_k(w_1) = v_1.$$

Другими словами, $[A \to \beta_1.\beta_2,\ u]$ — $LR(k)$-ситуация, допустимая для активного префикса γ.

Случай 2: данная $LR(k)$-ситуация получена посредством замыкания $[A_1 \to \alpha_1 X.\alpha_2,\ v_1] \in V_k^G(\gamma)$. Это значит, что согласно алгоритму данная $LR(k)$-ситуация $[A \to \beta_1.\beta_2,\ u]$ получается посредством нескольких шагов 2б, производящих последовательность $LR(k)$-ситуаций следующим образом:

$\alpha_2 = A_2\delta_2$, $\exists\ A_2 \to \alpha_3 \in P$, и шаг 2б дает $[A_2 \to .\alpha_3,\ v_2] \in V_k^G(\gamma)$,

где $v_2 \in \mathrm{FIRST}_k(\delta_2 v_1)$, т.е. $\exists\ \delta_2 \underset{\text{rm}}{\overset{*}{\Longrightarrow}} w_2$ и $v_2 = \mathrm{FIRST}_k(w_2 v_1)$;

$\alpha_3 = A_3\delta_3$, $\exists\ A_3 \to \alpha_4 \in P$, и шаг 2б дает $[A_3 \to .\alpha_4,\ v_3] \in V_k^G(\gamma)$,

где $v_3 \in \mathrm{FIRST}_k(\delta_3 v_2)$, т.е. $\exists\, \delta_3 \underset{\text{rm}}{\overset{*}{\Longrightarrow}} w_3$ и $v_3 = \mathrm{FIRST}_k(w_3 v_2) =$

$= \mathrm{FIRST}_k(w_3 w_2 v_1)$;

…

$\alpha_m = A_m \delta_m$, $\exists\, A_m \to \alpha_{m+1} \in P$, и шаг 2б дает

$[A_m \to .\alpha_{m+1},\ v_m] = [A \to \beta_1.\beta_2,\ u] \in V_k^G(\gamma)$,

где $v_m \in \mathrm{FIRST}_k(\delta_m v_{m-1})$, т.е. $\exists\, \delta_m \underset{\text{rm}}{\overset{*}{\Longrightarrow}} w_m$ и $v_m = \mathrm{FIRST}_k(w_m v_m) =$

$= \mathrm{FIRST}_k(w_m \ldots w_2 v_1)$.

Кроме того, из равенства двух $LR(k)$-ситуаций следует $A_m = A$, $\beta_1 = \varepsilon$, $\alpha_{m+1} = \beta_2$, $v_m = u$; также существует правило $A \to \beta_2$.

Извлечем теперь полезные следствия из вышеизложенной истории.

Из п.1 согласно индукционной гипотезе следует существование вывода вида

$S \underset{\text{rm}}{\overset{*}{\Longrightarrow}} \alpha_0 A_1 w_1 \underset{\text{rm}}{\Longrightarrow} \alpha_0 \alpha_1 X \alpha_2 w_1 = \gamma' X \alpha_2 w_1 = \gamma A_2 \delta_2 w_1$, в котором $\mathrm{FIRST}_k(w_1) = v_1$.

Благодаря п.3 этот вывод может быть продолжен следующим образом:

$S \underset{\text{rm}}{\overset{*}{\Longrightarrow}} \gamma A_2 \delta_2 w_1 \underset{\text{rm}}{\overset{*}{\Longrightarrow}} \gamma A_2 w_2 w_1 \underset{\text{rm}}{\Longrightarrow} \gamma \alpha_3 w_2 w_1 = \gamma A_3 \delta_3 w_2 w_1 \underset{\text{rm}}{\overset{*}{\Longrightarrow}} \gamma A_3 w_3 w_2 w_1 \underset{\text{rm}}{\overset{*}{\Longrightarrow}} \gamma A_m w_m \ldots w_1 =$

$= \gamma A w \underset{\text{rm}}{\Longrightarrow} \gamma \beta_2 w$, $w = w_m \ldots w_1$ и $\mathrm{FIRST}_k(w) = v_m = u$.

Согласно определению, учитывая вышеприведенный вывод, заключаем, что $LR(k)$-ситуация $[A \to \beta_1.\beta_2,\ u] \in V_k^G(\gamma)$ допустима для активного префикса γ. Утверждение II доказано.

Из рассуждений I и II следует справедливость теоремы.

Алгоритм 3.3: построение системы множеств допустимых $LR(k)$-ситуаций для данной КС-грамматики.

Вход: $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика.

Выход: $\mathcal{S} = \{\mathcal{A} \mid \mathcal{A} = V_k^G(\gamma)$, где γ — активный префикс $G\}$.

Метод.

1. Построить множество $\mathcal{A}_0 = V_k^G(\varepsilon)$ и поместить его в $\mathcal{S}$ как непомеченное множество.

2. Пусть $\mathcal{A} \in \mathcal{S}$ и $\mathcal{A}$ — не помечено. Пометить его и построить множество $\mathcal{A}' = \mathrm{GOTO}(\mathcal{A}, X)$ для всех $X \in V_N \cup V_T$. Если $\mathcal{A}' \neq \varnothing$ и $\mathcal{A}' \notin \mathcal{S}$, то включить $\mathcal{A}'$ в $\mathcal{S}$ как непомеченное множество.

3. Повторять шаг 2 до тех пор, пока все множества $LR(k)$-ситуаций в $\mathcal{S}$ не будут помечены. Момент, когда в $\mathcal{S}$ все множества окажутся помеченными, обязательно наступит, так как число правил грамматики G конечно, число позиций в них конечно, число терминальных цепочек, длина которых не превосходит k, конечно и соответственно число $LR(k)$-ситуаций, допустимых для грамматики G, тоже конечно.

4. Полученное множество $\mathcal{S}$ является искомым.

***Определение* 3.10.** Если G — контекстно-свободная грамматика, то систему множеств допустимых $LR(k)$-ситуаций для пополненной грамматики G' будем называть *канонической системой множеств $LR(k)$-ситуаций* для грамматики G.

Замечание 3.4. Множество GOTO($\mathscr{A}$, S') никогда не потребуется вычислять, так как оно всегда пусто[18].

Пример 3.10. Рассмотрим еще раз пополненную грамматику из примера 3.1, содержащую следующие правила: 0) $S' \to S$, 1) $S \to SaSb$, 2) $S \to \varepsilon$, и проиллюстрируем на ней алгоритм 3.3.

Как положено, начинаем с построения:

$\mathscr{A}_0 = V_1^G(\varepsilon) = \{[S' \to .S, \varepsilon], [S \to .SaSb, \varepsilon \mid a], [S \to ., \varepsilon \mid a]\}$ (см. пример 3.9).

Затем строим GOTO($\mathscr{A}_0, X$), где $X \in \{S, a, b\}$:

$\mathscr{A}_1$=GOTO($\mathscr{A}_0, S$)=$V_1^G(S)$=$\{[S' \to S., \varepsilon], [S \to S.aSb, \varepsilon \mid a]\}$ (см. пример 3.9);

GOTO($\mathscr{A}_0, a$) = GOTO($\mathscr{A}_0, b$) = $\varnothing$.

Теперь вычисляем GOTO($\mathscr{A}_1, X$), где $X \in \{S, a, b\}$:

GOTO($\mathscr{A}_1, S$) = $\varnothing$;

$\mathscr{A}_2$=GOTO($\mathscr{A}_1, a$)=GOTO($V_1^G(S), a$)=$V_1^G(Sa)$=

= $\{[S \to Sa.Sb, \varepsilon \mid a], [S \to .SaSb, a \mid b], [S \to ., a \mid b]\}$ (см. пример 3.9);

GOTO($\mathscr{A}_1, b$) = GOTO($V_1^G(S), b$) = $\varnothing$.

Теперь вычисляем GOTO($\mathscr{A}_2, X$), где $X \in \{S, a, b\}$:

$\mathscr{A}_3$=GOTO($\mathscr{A}_2, S$)=$\{[S \to SaS.b, \varepsilon \mid a], [S \to S.aSb, a \mid b]\}$;

GOTO($\mathscr{A}_2, a$)= GOTO($V_1^G(Sa), a$)=$\varnothing$;

GOTO($\mathscr{A}_2, b$)= GOTO($V_1^G(Sa), b$)=$\varnothing$.

Теперь вычисляем GOTO($\mathscr{A}_3, X$), где $X \in \{S, a, b\}$:

GOTO ($\mathscr{A}_3, S$)=$\varnothing$;

$\mathscr{A}_4$ = GOTO($\mathscr{A}_3, a$) = $\{[S \to Sa.Sb, a \mid b], [S \to .SaSb, a \mid b], [S \to ., a \mid b]\}$;

$\mathscr{A}_5$ = GOTO($\mathscr{A}_3, b$) = $\{[S \to SaSb., \varepsilon \mid a\}$.

Теперь вычисляем GOTO($\mathscr{A}_4, X$), где $X \in \{S, a, b\}$:

$\mathscr{A}_6$ = GOTO($\mathscr{A}_4, S$) = $\{[S \to SaS.b, a \mid b], [S \to S.aSb, a \mid b]\}$.

GOTO($\mathscr{A}_4, a$) = GOTO($\mathscr{A}_4, b$) = $\varnothing$.

Теперь вычисляем GOTO($\mathscr{A}_5, X$), где $X \in \{S, a, b\}$:

GOTO($\mathscr{A}_5, S$) = GOTO($\mathscr{A}_5, a$) = GOTO($\mathscr{A}_5, b$) = $\varnothing$.

Теперь вычисляем GOTO ($\mathscr{A}_6, X$), где $X \in \{S, a, b\}$:

GOTO($\mathscr{A}_6, S$) = $\varnothing$;

[18] Так как S' не встречается в правых частях правил.

$\mathrm{GOTO}(\mathscr{A}_6, a) = \{[S \to Sa.Sb, a \mid b], S \to .SaSb, a|b], [S \to ., a|b]\} = \mathscr{A}_4$;

$\mathscr{A}_7 = \mathrm{GOTO}(\mathscr{A}_6, b) = \{[S \to SaSb., a \mid b]\}$.

Таким образом $\{\mathscr{A}_0, \mathscr{A}_1, \mathscr{A}_2, \mathscr{A}_3, \mathscr{A}_4, \mathscr{A}_5, \mathscr{A}_6, \mathscr{A}_7\}$ — каноническая система множеств $LR(1)$-ситуаций для грамматики G.

Табл. 3.4 представляет функцию $\mathrm{GOTO}(\mathscr{A}, X)$ для грамматики G. Заметим, что с точностью до обозначений она совпадает с частью $g(X)$ табл. 3.3.

Таб. 3.4

$\mathscr{A}$	X		
	S	a	b
$\mathscr{A}_0$	$\mathscr{A}_1$		
$\mathscr{A}_1$		$\mathscr{A}_2$	
$\mathscr{A}_2$	$\mathscr{A}_3$		
$\mathscr{A}_3$		$\mathscr{A}_4$	$\mathscr{A}_5$
$\mathscr{A}_4$	$\mathscr{A}_6$		
$\mathscr{A}_5$			
$\mathscr{A}_6$		$\mathscr{A}_4$	$\mathscr{A}_7$
$\mathscr{A}_7$			

Пустые клетки в ней соответствуют неопределенным значениям. Заметим, что множество $\mathrm{GOTO}(\mathscr{A}, X)$ всегда пусто, если в каждой $LR(k)$-ситуации из $\mathscr{A}$ точка расположена на конце правила. Примерами таких множеств здесь служат $\mathscr{A}_5$ и $\mathscr{A}_7$.

Теорема 3.3. *Пусть $G = (V_\mathrm{N}, V_\mathrm{T}, P, S)$ — контекстно-свободная грамматика. Множество $LR(k)$-ситуаций $\mathscr{A} \in \mathscr{S}$ тогда и только тогда, когда существует активный префикс $\gamma \in (V_\mathrm{N} \cup V_T)^*$, такой, что $V_k^G(\gamma) = \mathscr{A}$.*

Доказательство.

I. Докажем сначала, что если $\mathscr{A} \in \mathscr{S}$, то существует активный префикс $\gamma \in (V_\mathrm{N} \cup V_\mathrm{T})^*$, такой, что $V_k^G(\gamma) = \mathscr{A}$.

Согласно алгоритму 3.3 элементы множества $\mathscr{S}$ образуются в определенной последовательности, т.е. $\mathscr{S} = \bigcup_{i=0}^{m} \mathscr{S}_i$, причем сначала строится множество $\mathscr{S}_0 = \{\mathscr{A}_0 \mid \mathscr{A}_0 = V_k^G(\varepsilon)\}$, а элементы из множества $\mathscr{S}_{i+1}$ строятся по элементам множества $\mathscr{S}_i$ следующим образом:

если $\mathscr{A}' = V_k^G(\gamma') \in \mathscr{S}_i$, то $\mathscr{A} = \mathrm{GOTO}(\mathscr{A}', X) = V_k^G(\gamma' X) \in \mathscr{S}_{i+1}$, где $X \in V_\mathrm{N} \cup V_\mathrm{T}$.

Доказательство проведем индукцией по номеру i множества $\mathscr{S}_i$, которому принадлежит элемент $\mathscr{A}$.

База. Пусть $i = 0$. Имеем $\mathscr{A} \in \mathscr{S}_0$. Согласно алгоритму 3.3 $\mathscr{A}_0 = V_k^G(\varepsilon)$, и по теореме 3.2 о корректности алгоритма построения функции $V_k^G(\gamma)$ цепочка $\gamma = \varepsilon$ — как раз такой активный префикс грамматики G, что $V_k^G(\varepsilon) = \mathscr{A}_0 = \mathscr{A} \in \mathscr{S}_0 \subseteq \mathscr{S}$.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех $i \le n$ ($n < m$).

Индукционный переход. Покажем, что аналогичное утверждение выполняется для $i = n + 1$. Пусть $\mathscr{A} \in \mathscr{S}_{n+1}$. В соответствии с алгоритмом 3.3 существуют $\mathscr{A}' \in \mathscr{S}_n$ и $X \in V_N \cup V_T$, такие, что $\mathscr{A} = \mathrm{GOTO}(\mathscr{A}', X) = V_k^G(\gamma' X) = V_k^G(\gamma)$, где $\gamma = \gamma' X$.

Согласно теореме 3.2 цепочка γ является активным префиксом грамматики G, таким, что $\mathscr{A} = V_k^G(\gamma)$. Утверждение I доказано.

II. Докажем теперь, что если $\gamma \in (V_N \cup V_T)^*$ — активный префикс грамматики G и $\mathscr{A} = V_k^G(\gamma)$, то $\mathscr{A} \in \mathscr{S}$.

Индукция по $l = |\gamma|$.

База. Пусть $l = 0$, $\gamma = \varepsilon$. Имеем $\mathscr{A} = V_k^G(\varepsilon) = \mathscr{A}_0$. Согласно алгоритму 3.3 $\mathscr{A}_0$ включается в множество $\mathscr{S}$ на первом же шаге.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех γ: $l = |\gamma| \le n$ ($n \ge 0$).

Индукционный переход. Покажем, что такое утверждение выполняется для $l = n + 1$. Пусть $\gamma = \gamma' X$, где $|\gamma'| = n$, $X \in V_N \cup V_T$ и γ — активный префикс грамматики G. Это значит, что существует правосторонний вывод вида

$$S \overset{*}{\underset{rm}{\Longrightarrow}} \alpha A w \underset{rm}{\Longrightarrow} \alpha\beta_1\beta_2 w = \gamma\beta_2 w = \gamma' X \beta_2 w, \text{ где } \alpha\beta_1 = \gamma = \gamma' X.$$

Следовательно, γ' — тоже активный префикс грамматики G и $|\gamma'| = n$. Согласно индукционной гипотезе $V_k^G(\gamma') = \mathscr{A}' \in \mathscr{S}$. Кроме того, в соответствии с алгоритму 3.3 $\mathscr{A} = \mathrm{GOTO}(\mathscr{A}', X) = \mathrm{GOTO}(V_k^G(\gamma'), X) = V_k^G(\gamma' X) = V_k^G(\gamma)$ включается в множество $\mathscr{S}$.

Утверждение II доказано. Из рассуждений I и II следует утверждение теоремы.

§ 3.5. Тестирование *LR*(*k*)-грамматик

Здесь мы рассмотрим метод проверки, является ли данная cfg $LR(k)$-грамматикой при заданном значении $k \ge 0$.

***Определение* 3.11.** Множество $\mathscr{A} = V_k^G(\gamma) \in \mathscr{S}$ назовем *непротиворечивым*, если оно не содержит двух разных $LR(k)$-ситуаций вида $[A \to \beta.,\ u]$ и $[B \to \beta_1.\beta_2,\ v]$ (цепочка β_2 может быть пустой), таких, что $u \in \mathrm{EFF}_k^G(\beta_2 v)$.

Алгоритм 3.4: $LR(k)$-тестирование.

Вход: $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика, $k \geq 0$.

Выход: "Да", если G — $LR(k)$-грамматика. "Нет" в противном случае.

Метод.

1. Построим каноническую систему $\mathscr{S}$ множеств $LR(k)$-ситуаций, допустимых для грамматики G.

2. Каждое $\mathscr{A} \in \mathscr{S}$ проверим на непротиворечивость. Если окажется, что рассматриваемое множество $\mathscr{A}$ противоречиво, то алгоритм заканчивается с ответом "Нет".

3. Если алгоритм не закончился на шаге 2, то он выдает ответ "Да" и завершается.

Замечание 3.5. При тестировании достаточно просматривать лишь множества $\mathscr{A} \in \mathscr{S}$, в которых имеется хотя бы одна $LR(k)$-ситуация вида $[A \to \beta., u]$ (с точкой в конце правила).

Пример 3.11. Протестируем грамматику примера 3.10, содержащую следующие правила: 0) $S' \to S$, 1) $S \to SaSb$, 2) $S \to \varepsilon$.

В соответствии с замечанием 3.5 необходимо и достаточно проверить лишь непротиворечивость множеств $\mathscr{A}_0$, $\mathscr{A}_1$, $\mathscr{A}_2$, $\mathscr{A}_4$, $\mathscr{A}_5$, $\mathscr{A}_7$, построенных в предыдущем примере (новая нумерация).

Проверка $\mathscr{A}_0 = \{[S' \to .S, \varepsilon], [S \to .SaSb, \varepsilon \mid a], [S \to ., \varepsilon \mid a]\}$:

$$u \in \{\varepsilon, a\},\ u \notin \mathrm{EFF}_1^G(S\{\varepsilon\}) = \varnothing;\ \ u \notin \mathrm{EFF}_1^G(SaSb\{\varepsilon, a\}) = \varnothing.$$

Вывод: множество $\mathscr{A}_0$ непротиворечиво.

Проверка $\mathscr{A}_1 = \{[S' \to S., \varepsilon], [S \to S.aSb, \varepsilon \mid a]\}$:

$$u = \varepsilon,\ u \notin \mathrm{EFF}_1^G(aSb\{\varepsilon, a\}) = \{a\}.$$

Вывод: множество $\mathscr{A}_1$ непротиворечиво.

Проверка $\mathscr{A}_2 = \{[S \to Sa.Sb, \varepsilon \mid a], [S \to .SaSb, a \mid b], [S \to ., a \mid b]\}$:

$$u \in \{a, b\},\ u \notin \mathrm{EFF}_1^G(Sb\{\varepsilon, a\}) = \varnothing,\ u \notin \mathrm{EFF}_1^G(SaSb\{a, b\}) = \varnothing.$$

Вывод: множество $\mathscr{A}_2$ непротиворечиво.

Проверка $\mathscr{A}_4 = \{[S \to Sa.Sb, a \mid b], [S \to .SaSb, a \mid b], [S \to ., a \mid b]\}$:

$$u \in \{a, b\},\ u \notin \mathrm{EFF}_1^G(Sb\{a, b\}) = \mathrm{EFF}_1^G(SaSb\{a, b\}) = \varnothing.$$

Вывод: множество $\mathscr{A}_4$ непротиворечиво.

Проверка $\mathscr{A}_5 = \{[S \to SaSb., \varepsilon \mid a]\}$ — множество непротиворечиво.

Проверка $\mathscr{A}_7 = \{[S \to SaSb., a \mid b]\}$ — множество непротиворечиво.

Заметим, что $\mathrm{EFF}_1^G(Sb) = \mathrm{EFF}_1^G(SaSb) = \varnothing$ потому, что из цепочек, начинающихся на нетерминал S, выводимы терминальные цепочки только благодаря ε-правилу, используемому на последнем шаге. Таким образом, получаемые цепочки не участвуют в образовании EFF_1^G.

Поскольку противоречивых множеств не найдено, то рассматриваемая грамматика — $LR(1)$-грамматика.

Теорема 3.4. *Алгоритм* 3.4 *дает правильный ответ, т.е. является правильным алгоритмом тестирования*.

Доказательство. Утверждение теоремы является прямым следствием теоремы 3.1, на которой и основывается алгоритм 3.4.

§ 3.6. Канонические *LR*(*k*)-анализаторы

***Определение* 3.12.** Пусть $G = (V_N, V_T, P, S)$ — контекстно-свободная грамматика и $\mathscr{S}$ — каноническое множество *LR*(*k*)-ситуаций для грамматики *G*. Для каждого множества $\mathscr{A} \in \mathscr{S}$ определим *LR*(*k*)-*таблицу* $T(\mathscr{A}) = (f, g)$, *состоящую из пары функций*:

f: $V_T^{*k} \to$ {shift, reduce *i*, accept, error},

g: $V_N \cup V_T \to \{T(\mathscr{A}) \mid \mathscr{A} \in \mathscr{S}\} \cup \{\text{error}\}$.

Функция f определяется следующим образом:

а) $f(u)$ = shift, если существует $[A \to \beta_1.\beta_2, v] \in \mathscr{A}$, $\beta_2 \neq \varepsilon$ и $u \in \mathrm{EFF}_k^G(\beta_2 v)$;

б) $f(u)$ = reduce *i*, $[A \to \beta., u] \in \mathscr{A}$ и $A \to \beta$ есть *i*-е правило из множества правил P, $i \geq 1$[19];

в) $f(u)$ = accept, если $[S' \to S., \varepsilon] \in \mathscr{A}$ и $u = \varepsilon$;

г) $f(u)$ = error в остальных случаях.

Если G — *LR*(*k*)-грамматика, то никаких неоднозначностей по пунктам а) и б) не может быть.

Функция g определяется следующим образом:

$$g(X) = \begin{cases} T(\mathscr{A}'), \text{ где } \mathscr{A}' = \mathrm{GOTO}(\mathscr{A}, X), \text{ если } \mathscr{A}' \neq \varnothing; \\ \text{error, если } \mathscr{A}' = \varnothing. \end{cases}$$

***Определение* 3.13.** Будем говорить, что *LR*(*k*)-*таблица* $T(\mathscr{A})$ *соответствует активному префиксу* γ, если $\mathscr{A} = V_k^G(\gamma)$.

***Определение* 3.14.** *Канонической системой LR*(*k*)-*таблиц для* cfg G назовем пару $(\mathscr{T}, T_0)$, где $\mathscr{T}$ — множество *LR*(*k*)-таблиц, соответствующих канонической системе множеств *LR*(*k*)-ситуаций для G, а $T_0 = T(\mathscr{A}_0)$, где $\mathscr{A}_0 = V_k^G(\varepsilon)$.

Обычно *LR*(*k*)-анализатор представляется управляющей таблицей, каждая строка которой является *LR*(*k*)-таблицей.

***Определение* 3.15.** Описанный ранее алгоритм 3.1 (см. § 3.3), если он использует каноническую систему *LR*(*k*)-таблиц, назовем *каноническим LR*(*k*)-*алгоритмом разбора* или просто *каноническим LR*(*k*)-*анализатором*.

---

[19] Напомним, что под нулевым номером числится правило $S' \to S$, пополняющее данную грамматику G.

Алгоритм 3.5: построение канонического $LR(k)$-анализатора.

Вход: $G = (V_N, V_T, P, S)$ — $LR(k)$-грамматика, $k \geq 0$.

Выход: $(\mathcal{T}, T_0)$ — каноническая система $LR(k)$-таблиц для грамматики G.

Метод.

1. Построить каноническую систему множеств $LR(k)$-ситуаций $\mathcal{S}$ для грамматики G.

2. Взять $\mathcal{T} = \{T(\mathcal{A}) \mid \mathcal{A} \in \mathcal{S}\}$ в качестве множества $LR(k)$-таблиц.

3. Положить $T_0 = T(\mathcal{A}_0)$, где $\mathcal{A}_0 = V_k^G(\varepsilon)$.

Пример 3.12. Построим каноническую систему $LR(k)$-таблиц для грамматики из примера 3.10, содержащей следующие правила: 0) $S' \to S$, 1) $S \to SaSb$, 2) $S \to \varepsilon$, учитывая результаты построения системы множеств $LR(k)$-ситуаций и функции GOTO, приведенные в этом примере.

Поскольку $\mathcal{A}_0 = \{[S' \to .S, \varepsilon], [S \to .SaSb, \varepsilon \mid a], [S \to ., \varepsilon \mid a]\}$, то $T_0 = (f_0, g_0)$ имеет следующий табличный вид:

$f_0(u)$			$g_0(X)$		
a	b	ε	S	a	b
reduce 2	error	reduce 2	T_1	error	error

Поскольку $\mathcal{A}_1 = \{[S' \to S., \varepsilon], [S \to S.aSb, \varepsilon \mid a]\}$, то $T_1 = (f_1, g_1)$ имеет следующий табличный вид:

$f_1(u)$			$g_1(X)$		
a	b	ε	S	a	b
shift	error	accept	error	T_2	error

Поскольку $\mathcal{A}_2 = \{[S \to Sa.Sb, \varepsilon \mid a], [S \to .SaSb, a \mid b], [S \to ., a \mid b]\}$, то $T_2 = (f_2, g_2)$ имеет следующий табличный вид:

$f_2(u)$			$g_2(X)$		
a	b	ε	S	a	b
reduce 2	reduce 2	error	T_3	error	error

Поскольку $\mathcal{A}_3 = \{[S \to SaS.b, \varepsilon \mid a], [S \to S.aSb, a \mid b]\}$, то $T_3 = (f_3, g_3)$ имеет следующий табличный вид:

$f_3(u)$			$g_3(X)$		
a	b	ε	S	a	b
shift	shift	error	error	T_4	T_5

Поскольку $\mathcal{A}_4 = \{[S \to Sa.Sb, a \mid b], [S \to .SaSb, a \mid b], [S \to ., a \mid b]\}$, то $T_4 = (f_4, g_4)$ имеет следующий табличный вид:

$f_4(u)$			$g_4(X)$		
a	b	ε	S	a	b
reduce 2	reduce 2	error	T_6	error	error

Поскольку $\mathcal{A}_5 = \{[S \to SaSb., \varepsilon \mid a\}$, то $T_5 = (f_5, g_5)$ имеет следующий табличный вид:

$f_5(u)$			$g_5(X)$		
a	b	ε	S	a	b
reduce 1	error	reduce 1	error	error	error

Поскольку $\mathcal{A}_6 = \{[S \to SaS.b,\ a \mid b],\ [S \to S.aSb,\ a \mid b]\}$, то $T_6 = (f_6, g_6)$ имеет следующий табличный вид:

$f_6(u)$			$g_6(X)$		
a	b	ε	S	a	b
shift	shift	error	error	T_4	T_7

Поскольку $\mathcal{A}_7 = \{[S \to SaSb.,\ a \mid b]\}$, то $T_7 = (f_7, g_7)$ имеет следующий табличный вид:

$f_7(u)$			$g_7(X)$		
a	b	ε	S	a	b
reduce 1	reduce 1	error	error	error	error

Все эти $LR(k)$-таблицы сведены в управляющую таблицу 3.1 канонического $LR(k)$-анализатора, которая была приведена в примере 3.6.

Итак, $LR(k)$-анализаторы обладают следующими свойствами:

1. Диагностичностью. Простой индукцией по числу шагов анализатора можно показать, что каждая $LR(k)$-таблица, находящаяся в его магазине, соответствует цепочке символов, расположенной слева от таблицы. Именно: если в цепочке, о которой идет речь, отбросить символы $LR(k)$-таблиц, то получим активный префикс грамматики, которому эта $LR(k)$-таблица соответствует. Поэтому $LR(k)$-анализатор сообщает об ошибке при первой же возможности в ходе чтения входной цепочки слева направо.

2. Пусть $T_j = (f_j, g_j)$. Если $f_j(u) = \text{shift}$ и анализатор находится в конфигурации $(T_0X_1T_1X_2...X_jT_j,\ x,\ \pi)$, то найдется $LR(k)$-ситуация $[B \to \beta_1.\beta_2,\ v]$, $\beta_2 \neq \varepsilon$, допустимая для активного префикса $X_1X_2...X_j$, такая, что $u \in \text{EFF}_k^G(\beta_2 v)$, где $u = \text{FIRST}_k(x)$. Поэтому, если $S' \underset{\text{rm}}{\overset{*}{\Longrightarrow}} X_1X_2\ldots X_juy$ при некоторой цепочке $y \in V_\text{T}^*$, то по теореме 3.1 правый конец основы цепочки $X_1X_2...X_juy$ должен быть где-то справа от символа X_j.

3. Если в конфигурации, указанной в п.2, $f_j(u) = \text{reduce } i$ и $A \to Y_1Y_2...Y_p$ — правило с номером i, то цепочка $X_{j-p+1}...X_{j-1}X_j$ в магазине должна совпадать с $Y_1Y_2...Y_p$, так как множество ситуаций, по которому построена таблица T_j, содержит ситуацию $[A \to Y_1Y_2...Y_p.,\ u]$. Таким образом, на шаге свертки не обязательно рассматривать символы верхней части магазина, надо просто выбросить из магазина $2p$ символов грамматики и $LR(k)$-таблиц.

4. Если $f_j(u) = \text{accept}$, то $u = \varepsilon$. Содержимое магазина в этот момент имеет вид: T_0ST_1, где T_1 — $LR(k)$-таблица, соответствующая множеству $LR(k)$-ситуаций, в которое входит ситуация $[S' \to S., \varepsilon]$.

§ 3.7. Корректность *LR*(*k*)-анализаторов

Теорема 3.5. *Пусть* $G = (V_N, V_T, P, S)$ — *LR*(*k*)-*грамматика и пусть* $\mathfrak{A}$ — *LR*(*k*)-*анализатор, построенный по грамматике G согласно алгоритму* 3.5. *Утверждается, что вывод* $S \underset{rm}{\overset{\pi}{\Longrightarrow}} w$ *существует тогда и только тогда, когда* $(T_0, w, \varepsilon) \overset{*}{\underset{\mathfrak{A}}{\vdash}} (T_0 S T, \varepsilon, \pi^R)$, где $T = (f, g) \in \mathcal{T}$, *причем* $f(\varepsilon) =$ accept.

Доказательство.

I. Индукцией по $l = |\pi|$ покажем теперь, что если $S \underset{rm}{\overset{\pi}{\Longrightarrow}} \alpha x$, где $\alpha \in (V_N \cup V_T)^*$, $x \in V_T^*$, то $(T_0 X_1 T_1 X_2 \ldots X_m T_m, x, \varepsilon) \overset{*}{\underset{\mathfrak{A}}{\vdash}} (T_0 S T, \varepsilon, \pi^R)$, где $X_1 X_2 \ldots X_m = \alpha$, $T_0 = T(\mathscr{A}_0)$, $\mathscr{A}_0 = V_k^G(\varepsilon)$, а $T_i = T(\mathscr{A}_i)$, $\mathscr{A}_i = V_k^G(X_1 X_2 \ldots X_i)$, $i = 1, 2, \ldots, m$.

База. Пусть $l = 1$, т.е. $\pi = i$. Имеем $S \underset{rm}{\overset{(i)}{\Longrightarrow}} \alpha x$. Следовательно, существует правило (i) $S \to \alpha x \in P$.

Пусть $x = a_1 a_2 \ldots a_p$, $a_j \in V_T$, $j = 1, 2, \ldots, p$. Согласно определению активных префиксов и допустимых для них *LR*(*k*)-ситуаций имеем

$$[S \to \alpha . a_1 a_2 \ldots a_p, \varepsilon] \in \mathscr{A}_m = V_k^G(\alpha) = V_k^G(X_1 X_2 \ldots X_m),$$

$$[S \to \alpha a_1 . a_2 \ldots a_p, \varepsilon] \in \mathscr{A}_{m+1} = V_k^G(\alpha a_1),$$

$\ldots$

$$[S \to \alpha a_1 a_2 \ldots a_{p-1} . a_p, \varepsilon] \in \mathscr{A}_{m+p-1} = V_k^G(\alpha a_1 a_2 \ldots a_{p-1}),$$

$$[S \to \alpha a_1 a_2 \ldots a_p ., \varepsilon] = [S \to \alpha x ., \varepsilon] \in \mathscr{A}_{m+p} = V_k^G(\alpha a_1 a_2 \ldots a_p) = V_k^G(\alpha x).$$

Тогда согласно алгоритму построения *LR*(*k*)-таблиц $T_{m+j} = T(\mathscr{A}_{m+j}) = (f_{m+j}, g_{m+j})$, $f_{m+j-1}(u_j) =$ shift для $u_j \in \mathrm{EFF}_k^G(a_j a_{j+1} \ldots a_p \varepsilon) = \mathrm{FIRST}_k(a_j a_{j+1} \ldots a_p)$; $g_{m+j-1}(a_j) = T_{m+j}$ $(j = 1, 2, \ldots, p)$; а $T_{m+p} = (f_{m+p}, g_{m+p})$ и $f_{m+p}(\varepsilon) =$ reduce i. Поэтому

$$(T_0 X_1 T_1 X_2 \ldots X_m T_m, x, \varepsilon) = (T_0 X_1 T_1 X_2 \ldots X_m T_m, a_1 a_2 \ldots a_p, \varepsilon) \overset{*}{\underset{\mathfrak{A}}{\vdash}}$$
$$\overset{*}{\underset{\mathfrak{A}}{\vdash}} (T_0 X_1 T_1 X_2 \ldots X_m T_m a_1 T_{m+1} a_2 T_{m+2} \ldots a_p T_{m+p}, \varepsilon, \varepsilon) \underset{\mathfrak{A}}{\vdash} (T_0 S T, \varepsilon, i).$$

Здесь сначала выполняется p раз сдвиг, затем один раз свертка по i-му правилу; $X_1 X_2 \ldots X_m a_1 a_2 \ldots a_p = \alpha x$, $T = T(V_k^G(S)) = (f, g)$, причем очевидно, что $[S' \to S., \varepsilon] \in V_k^G(S)$ и потому согласно алгоритму 3.5 $f(\varepsilon) =$ accept. База доказана.

Индукционная гипотеза. Предположим, что утверждение I выполняется для всех $l \le n$ $(n \ge 1)$.

Индукционный переход. Покажем, что утверждение I выполняется для $l = n + 1$.

Имеем $S \underset{rm}{\overset{\pi'}{\Longrightarrow}} \alpha' A w \underset{rm}{\overset{(i)}{\Longrightarrow}} \alpha' \beta w = \alpha x$. Очевидно, что x заканчивается цепочкой w, т.е. $x = zw$. Здесь $\alpha', \alpha \in (V_N \cup V_T)^*$, $w, x, z \in V_T^*$, $\pi = \pi' i$, $A \to \beta \in P$ — (i)-е правило. Кроме того, $\alpha' \beta w = \alpha x = \alpha z w$, и потому $\alpha' \beta = \alpha z$.

Рассмотрим исходную конфигурацию

$$\overbrace{\overbrace{(T_0X_1T_1X_2\ldots X_jT_j}^{\alpha}\overbrace{X_{j+1}T_{j+1}\ldots X_mT_m}^{\beta'}\ ,\ \overbrace{zw}^{x},\ \varepsilon)}^{\alpha}\ .$$

$$\beta$$

В ней $\alpha' = X_1X_2\ldots X_j$, $\beta' = X_{j+1}\ldots X_m$, $\beta = \beta' z$, $\alpha = \alpha'\beta'$, $x = zw$. Поэтому имеющийся вывод представим в виде $S \underset{\text{rm}}{\overset{\pi'}{\Longrightarrow}} \alpha' A w \underset{\text{rm}}{\overset{(i)}{\Longrightarrow}} \alpha'\beta w = \alpha'\beta' zw$.

Поскольку $A \to \beta = \beta' z \in P$, то $\alpha'\beta'$ — активный префикс правосентенциальной формы $\alpha'\beta' zw$, причем $V_k^G(\alpha'\beta') = V_k^G(\alpha) = \mathscr{A}_m$, и потому $[A \to \beta'.z,\ u] \in \mathscr{A}_m$, где $u = \text{FIRST}_k(w)$. Пусть $z = a_1a_2\ldots a_p$. Согласно алгоритму построения $LR(k)$-таблиц имеем

$T_m = T(\mathscr{A}_m) = (f_m, g_m)$,

$f_m(v_1) = $ shift для $v_1 \in \text{EFF}_k^G(zu) = \text{FIRST}_k(a_1a_2\ldots a_pw) = \text{FIRST}_k(x)$,

$g_m(a_1) = T_{m+1}$, где $T_{m+1} = T(\mathscr{A}_{m+1})$, $\mathscr{A}_{m+1} = V_k^G(\alpha'\beta' a_1)$;

$T_{m+1} = (f_{m+1}, g_{m+1})$,

$f_{m+1}(v_2) = $ shift для $v_2 \in \text{FIRST}_k(a_2\ldots a_pu) = \text{FIRST}_k(a_2\ldots a_pw)$,

$g_{m+1}(a_2) = T_{m+2}$, где $T_{m+2} = T(\mathscr{A}_{m+2})$, $\mathscr{A}_{m+2} = V_k^G(\alpha'\beta' a_1a_2)$;

...

$T_{m+p-1} = T(\mathscr{A}_{m+p-1}) = (f_{m+p-1}, g_{m+p-1})$, $\mathscr{A}_{m+p-1} = V_k^G(\alpha'\beta' a_1a_2\ldots a_{p-1})$,

$f_{m+p-1}(v_p) = $ shift для $v_p \in \text{EFF}_k^G(a_pu) = \text{FIRST}_k(a_pu) = \text{FIRST}_k(a_pw)$,

$g_{m+p-1}(a_p) = T_{m+p} = T(\mathscr{A}_{m+p})$, где $\mathscr{A}_{m+p} = V_k^G(\alpha'\beta' a_1a_2\ldots a_{p-1}a_p) = V_k^G(\alpha'\beta' z) =$

$= V_k^G(\alpha'\beta)$.

$T_{m+p} = (f_{m+p}, g_{m+p})$, $f_{m+p}(u) = $ reduce i, $(i)\ A \to \beta \in P$, $u = \text{FIRST}_k(w)$.

Используя существующие $LR(k)$-таблицы, канонический $LR(k)$-анализатор совершает следующие движения, начиная с исходной конфигурации:

$(T_0X_1T_1X_2\ldots X_mT_m, x, \varepsilon) = (T_0X_1T_1X_2\ldots X_mT_m, zw, \varepsilon) =$

$= (T_0X_1T_1X_2\ldots X_mT_m, a_1a_2\ldots a_pw, \varepsilon) \vdash_{\mathfrak{A}}$

$\vdash_{\mathfrak{A}} (T_0X_1T_1X_2\ldots X_mT_ma_1T_{m+1}, a_2\ldots a_pw, \varepsilon) \vdash_{\mathfrak{A}} \ldots$

$\ldots \vdash_{\mathfrak{A}} (T_0X_1T_1X_2\ldots X_mT_ma_1T_{m+1}a_2\ldots a_pT_{m+p}, w, \varepsilon) =$

$= (T_0X_1T_1X_2\ldots X_jT_jX_{j+1}T_{j+1}\ldots X_mT_ma_1T_{m+1}a_2\ldots a_pT_{m+p}, w, \varepsilon) \vdash_{\mathfrak{A}}$

$\vdash_{\mathfrak{A}} (T_0X_1T_1X_2\ldots X_jT_jA^{T'_{j+1}}, w, i) \overset{*}{\vdash}_{\mathfrak{A}} (T_0ST, \varepsilon, i\pi'^R) = (T_0ST, \varepsilon, \pi^R)$.

Последний переход обоснован индукционной гипотезой с учетом того, что $T'_{j+1} = T(\mathcal{A}'_{j+1}), \mathcal{A}'_{j+1} = V_k^G(\alpha' A)$, $\alpha' = X_1 X_2 \ldots X_j$. Вспомогательное утверждение I доказано.

В частности, если $\alpha = \varepsilon$, т.е. $S \underset{\text{rm}}{\overset{\pi}{\Longrightarrow}} x$, то $(T_0, x, \varepsilon) \overset{*}{\underset{\text{g}}{\vdash}} (T_0 S T, \varepsilon, \pi^R)$. Достигнутая конфигурация является принимающей по тем же причинам, что и в базовом случае.

II. Индукцией по l — числу движений $LR(k)$-анализатора — покажем, что если $(T_0, w, \varepsilon) \overset{l}{\underset{\text{g}}{\vdash}} (T_0 X_1 T_1 X_2 \ldots X_m T_m, x, \pi^R)$, то $\alpha x \underset{\text{rm}}{\overset{\pi}{\Longrightarrow}} w$, где $\alpha = X_1 X_2 \ldots X_m$, $X_j \in V_\text{N} \cup V_\text{T}$, $j = 1, 2, \ldots, m$; $x, w \in V_\text{T}^*$. Другими словами, покажем, что магазинная цепочка (без учета $LR(k)$-таблиц) и конкатенированная с ней входная цепочка представляют правосентенциальную форму, из которой выводится анализируемое предложение посредством последовательности правил π.

База. Пусть $l = 1$: имеется одно движение.

Случай 1: shift-движение.

Пусть $(T_0, w, \varepsilon) = (T_0, X_1 x, \varepsilon) \underset{\text{g}}{\vdash} (T_0 X_1 T, x, \varepsilon)$. Здесь $w = X_1 x$, где $X_1 \in V_\text{T}$, $x \in V_\text{T}^*$. Первый символ цепочки w переносится в магазин, на вершине оказывается некоторая $LR(k)$-таблица T, в выходную цепочку ничего не пишется. Следует показать, что существует вывод $X_1 x \underset{\text{rm}}{\overset{*}{\Longrightarrow}} w$ без использования каких-либо правил грамматики. Но это возможно лишь в случае, когда $w = X_1 x$, что как раз и имеет место.

Случай 2: reduce i -движение.

Это движение предполагает свертку какой-то цепочки в верхней части магазина в нетерминал по правилу номер i. Поскольку в начальной конфигурации (T_0, w, ε) магазин пуст (T_0 не считается), то основа магазинной цепочки тоже пуста. Именно она и сворачивается к некоторому нетерминалу. Формально это движение обусловлено тем, что $T_0 = (f_0, g_0)$, $f_0(u) = \text{reduce } i$ для $u = \text{FIRST}_k(w)$.

Наряду с этим согласно алгоритму построения $LR(k)$-анализатора $T_0 = T(\mathcal{A}_0)$, $\mathcal{A}_0 = V_k^G(\varepsilon)$, $[A \to \varepsilon., u] \in \mathcal{A}_0$, что и обусловливает движение:

$$(T_0, w, \varepsilon) \underset{\text{g}}{\vdash} (T_0 A T, w, \varepsilon), \text{ где } T = g_0(A).$$

Остается проверить, что $Aw \underset{\text{rm}}{\overset{\{i\}}{\Longrightarrow}} w$. Но это очевидно, поскольку данное движение обусловлено существованием правила $A \to \varepsilon \in P$, номер которого есть i. База доказана.

Индукционная гипотеза. Предположим, что утверждение II выполняется для всех $l \le n$ $(n \ge 1)$.

Индукционный переход. Покажем, что утверждение II выполняется для $l = n + 1$. Пусть первые n движений дают $(T_0, w, \varepsilon) \overset{n}{\underset{\text{g}}{\vdash}} (T_0 X_1 T_1 X_2 \ldots X_m T_m, x', \pi'^R)$. По индукционной гипотезе немедленно получаем $X_1 X_2 \ldots X_m x' \underset{\text{rm}}{\overset{\pi'}{\Longrightarrow}} w$. Затем совершается последнее, $(n + 1)$-е, движение.

Случай 1: shift-движение.

Это движение происходит следующим образом:

$(T_0X_1T_1X_2\ldots X_mT_m,\ x',\ \pi'^R) =$

$= (T_0X_1T_1X_2\ldots X_mT_m, X_{m+1}x, \pi'^R) \vdash_{\mathfrak{g}} (T_0X_1T_1X_2\ldots X_mT_mX_{m+1}T_{m+1}, x, \pi'^R),$

т.е. X_{m+1} переносится в магазин, в выходную цепочку ничего не пишется. Остается лишь убедиться в том, что $X_1X_2\ldots X_mX_{m+1}x \underset{\text{rm}}{\overset{\pi'}{\Longrightarrow}} w$. Так как $x' = X_{m+1}x$, то $X_1X_2\ldots X_mX_{m+1}x = X_1X_2\ldots X_mx' \underset{\text{rm}}{\overset{\pi'}{\Longrightarrow}} w$ по индукционной гипотезе.

Случай 2: reduce i -движение.

Имеем конфигурацию, достигнутую за первые n движений: $(T_0X_1T_1X_2\ldots X_mT_m, x', \pi'^R)$. Далее совершается последнее движение: свертка верхней части магазина по i-у правилу. Оно происходит благодаря тому, что $T_m = (f_m,\ g_m)$, $f_m(u') =$ reduce i для $u' = \text{FIRST}_k(x')$.

Согласно алгоритму построения анализатора $T_m = T(\mathcal{A}_m)$, $\mathcal{A}_m = V_k^G(X_1X_2\ldots X_m)$, и существуют $LR(k)$-ситуация $[A \to Y_1Y_2\ldots Y_p., u'] \in \mathcal{A}_m$ и правило $A \to Y_1Y_2\ldots Y_p$ под номером i, такое, что $Y_1Y_2\ldots Y_p = X_{m-p+1}\ldots X_{m-1}X_m$. Имеем:

$(T_0X_1T_1X_2\ldots X_mT_m, x', \pi'^R) =$

$= (T_0X_1T_1X_2\ldots X_{m-p}T_{m-p}Y_1T_{m-p+1}Y_2T_{m-p+2}\ldots Y_pT_mx', \pi'^R) \vdash_{\mathfrak{g}}$

$\vdash_{\mathfrak{g}} (T_0X_1T_1X_2\ldots X_{m-p}T_{m-p}AT'_{m-p+1}, x', \pi'^R i),$

где $T'_{m-p+1} = g_{m-p}(A)$, если $T_{m-p} = (f_{m-p}, g_{m-p})$.

Остается убедиться лишь в том, что $X_1X_2\ldots X_{m-p}Ax' \underset{\text{rm}}{\overset{i\pi'}{\Longrightarrow}} w$. Действительно,

$X_1X_2\ldots X_{m-p}Ax' \underset{\text{rm}}{\overset{i}{\Longrightarrow}} X_1X_2\ldots X_{m-p}Y_1Y_2\ldots Y_px' =$

$= \text{X}_1\text{X}_2\ldots\text{X}_{\text{m}-\text{p}}\text{X}_{\text{m}-\text{p}+1}\ldots\text{X}_{\text{m}-1}\text{X}_{\text{m}}\text{x}' \underset{\text{rm}}{\overset{\pi'}{\Longrightarrow}} \text{w},$

причем первый шаг вывода выполняется при помощи упомянутого правила, а завершение вывода обеспечено индукционной гипотезой. Вспомогательное утверждение II доказано.

В частности, при $\alpha = S$ и $x = \varepsilon$ заключаем, что если $(T_0, w, \varepsilon) \overset{*}{\vdash}_{\mathfrak{g}} (T_0ST, \varepsilon, \pi^R)$, где последняя конфигурация — принимающая, то $S \underset{\text{rm}}{\overset{\pi}{\Longrightarrow}} w$.

Из рассуждений I и II следует утверждение теоремы.

Теорема 3.6. *Если $G = (V_N, V_T, P, S)$ — $LR(k)$-грамматика, то грамматика G — синтаксически однозначна.*

Доказательство ведется от противного. Пусть G — $LR(k)$-грамматика, но она не является синтаксически однозначной. Это значит, что существуют два разных правосторонних вывода одной и той же терминальной цепочки:

1) $S = \alpha_0 \underset{\text{rm}}{\Longrightarrow} \alpha_1 \underset{\text{rm}}{\Longrightarrow} \ldots \underset{\text{rm}}{\Longrightarrow} \alpha_m = w$,

2) $S = \beta_0 \underset{\text{rm}}{\Longrightarrow} \beta_1 \underset{\text{rm}}{\Longrightarrow} \ldots \underset{\text{rm}}{\Longrightarrow} \beta_n = w,\ w \in V_T^*$,

и существует $i > 0$ ($i < m$, $i < n$), такое, что $\alpha_{m-i} \neq \beta_{n-i}$, но $\alpha_{m-j} = \beta_{n-j}$ при $j < i$.

В более детальном представлении эти выводы имеют вид

$1'$) $S \underset{rm}{\overset{*}{\Longrightarrow}} \underbrace{\alpha A x}_{\alpha_{m-i}} \underset{rm}{\Longrightarrow} \alpha\beta x \underset{rm}{\overset{*}{\Longrightarrow}} w$,

$2'$) $S \underset{rm}{\overset{*}{\Longrightarrow}} \underbrace{\gamma B y}_{\beta_{n-i}} \underset{rm}{\Longrightarrow} \gamma\delta y = \underbrace{\gamma\overbrace{\beta_1}^{\delta}}_{\alpha\beta}\underbrace{\beta_2 y}_{x} = \alpha\beta x \underset{rm}{\overset{*}{\Longrightarrow}} w$, где $\delta = \beta_1\beta_2$, $\gamma\beta_1 = \alpha\beta$, $\beta_2 y = x$, $\beta_2 \in V_T^*$,

причем так как $\alpha_{m-i} \neq \beta_{n-i}$, то $\alpha \neq \gamma \vee A \neq B \vee x \neq y$.

Из вывода $1'$) очевидно, что $[A \to \beta., u] \in V_k^G(\alpha\beta)$, где $u = \mathrm{FIRST}_k(x)$.

Аналогично из $2'$) следует, что $[B \to \beta_1.\beta_2, v] \in V_k^G(\alpha\beta)$, $v = \mathrm{FIRST}_k(y)$. При этом

$$u = \mathrm{FIRST}_k(x) = \mathrm{FIRST}_k(\beta_2 y) = \mathrm{FIRST}_k(\beta_2) \oplus_k \mathrm{FIRST}_k(y) =$$
$$= \mathrm{FIRST}_k(\beta_2) \oplus_k \{v\} = \mathrm{FIRST}_k(\beta_2 v) = \mathrm{EFF}_k^G(\beta_2 v).$$

Последнее равенство имеет место, так как $\beta_2 \in V_T^*$. Существование этих двух $LR(k)$-ситуаций, допустимых для одного и того же активного префикса $\alpha\beta$, означает нарушение необходимого и достаточного $LR(k)$-условия, что противоречит исходному предположению о том, что G — $LR(k)$-грамматика. Следова-тельно теорема доказана.

Теорема 3.7. *Пусть* $G = (V_N, V_T, P, S)$ — $LR(k)$-*грамматика*. *Канонический* $LR(k)$-*анализатор*, *построенный по* G, *выполняя разбор цепочки* $x \in L(G)$, *совер-шает* $O(n)$ *движений*, *где* $n = |x|$.

Доказательство. Разбирая цепочку $x \in L(G)$, $LR(k)$-анализатор выполняет чередующиеся движения типа сдвиг (shift) и свертка (reduce). Любое из этих движений на вершине магазина устанавливает некоторую $LR(k)$-таблицу. Очевидно, что число сдвигов в точности равно n. Каждому сдвигу может предшествовать не более $\#\mathcal{T}$ сверток, где $\mathcal{T}$ — каноническое множество $LR(k)$-таблиц (состояний) анализатора. В противном случае какая-то из $LR(k)$-таблиц появилась бы повторно. При неизменной непросмотренной части входной цепочки это означало бы зацикливание анализатора. Тогда вследствие теоремы 3.5 существовало бы как угодно много правосторонних выводов сколь угодно большой длины одной и той же цепочки w, что означало бы неоднозначность грамматики G. На основании теоремы 3.6 грамматика G не являлась бы $LR(k)$-грам-матикой, что противоречило бы первоначальному предположению.

Итак, общее число движений канонического $LR(k)$-анализатора $N \le n + c \times \times n = n \times (c + 1) = O(n)$, где $c \le \#\mathcal{T}$; c — константа, зависящая от грамматики. Что и требовалось доказать.

Замечание 3.6. $LR(k)$-анализатор на ошибочных цепочках “зациклиться” не может. Цепочка — ошибочная, если для некоторого ее префикса не существует продолжения, дающего цепочку из $L(G)$. Действительно, если бы анализатор зациклился, прочитав только часть входной цепочки, то, как мы только что выяснили, это означало бы, что грамматика G не есть $LR(k)$-грамматика.

§ 3.8. Простые постфиксные синтаксически управляемые *LR*-трансляции

Мы знаем, что простые семантически однозначные схемы синтаксически управляемых трансляций с входными $LL(k)$-грамматиками определяют трансляции, реализуемые детерминированными магазинными преобразователями. Аналогичную ситуацию интересно рассмотреть в отношении схем с $LR(k)$-грамматиками в качестве входных. Но к сожалению, существуют такие простые семантически однозначные схемы, которые задают трансляции, не реализуемые детерминированными магазинными преобразователями.

Пример 3.13. Пусть имеется схема синтаксически управляемой трансляции

$$T = (\{S\},\ \{a, b\},\ \{a, b\},\ \{1)\ S \to Sa,\ aSa,\ \ 2)\ S \to Sb,\ bSb,\ \ 3)\ S \to \varepsilon,\ \varepsilon\}).$$

Очевидно, что ее входная грамматика — $LR(1)$. Действительно, каноническая система множеств допустимых $LR(1)$-ситуаций для этой грамматики $\mathscr{S} = \{\mathscr{A}_0, \mathscr{A}_1, \mathscr{A}_2, \mathscr{A}_3\}$ — не противоречива, ибо

$\mathscr{A}_0 = \{[S' \to .S, \varepsilon],\ [S \to .Sa, \varepsilon \mid a \mid b],\ [S \to .Sb, \varepsilon \mid a \mid b],\ [S \to ., \varepsilon \mid a \mid b]\}$,

$\mathscr{A}_1 = \mathrm{GOTO}(\mathscr{A}_0, S) = \{[S' \to S., \varepsilon],\ [S \to S.a, \varepsilon \mid a \mid b],\ [S \to S.b, \varepsilon \mid a \mid b]\}$,

$\mathscr{A}_2 = \mathrm{GOTO}(\mathscr{A}_1, a) = \{[S \to Sa., \varepsilon \mid a \mid b]\}$,

$\mathscr{A}_3 = \mathrm{GOTO}(\mathscr{A}_1, b) = \{[S \to Sb., \varepsilon \mid a \mid b]\}$.

Этому множеству соответствует управляющая таблица $LR(1)$-анализатора (табл. 3.5).

Табл. 3.5

T	$f(u)$			$g(X)$		
	a	b	ε	S	a	b
T_0	reduce 3	reduce 3	reduce 3	T_1	error	error
T_1	shift	shift	accept	error	T_2	T_3
T_2	reduce 1	reduce 1	reduce 1	error	error	error
T_3	reduce 2	reduce 2	reduce 2	error	error	error

Построенный канонический $LR(1)$-анализатор для входной цепочки bba выдает правосторонний анализ $\pi^R(bba) = 3221$, т.е.

$$(T_0, bba, \varepsilon) \vdash_{\text{я}} (T_0ST_1, bba, 3) \vdash_{\text{я}} (T_0ST_1bT_3, ba, 3) \vdash_{\text{я}} (T_0ST_1, ba, 32) \vdash_{\text{я}}$$
$$\vdash_{\text{я}} (T_0ST_1bT_3, a, 322) \vdash_{\text{я}} (T_0ST_1, a, 322) \vdash_{\text{я}} (T_0ST_1aT_2, \varepsilon, 3221) \vdash_{\text{я}} (T_0ST_1, \varepsilon, 3221).$$

Данная схема задает трансляцию

$$\tau(T) = \{(w, w^R w) \mid w \in \{a, b\}^*\}.$$

В частности, имеем вывод

$$(S, S) \overset{(1)}{\underset{\text{rm}}{\Longrightarrow}} (Sa, aSa) \overset{(2)}{\underset{\text{rm}}{\Longrightarrow}} (Sba, abSba) \overset{(2)}{\underset{\text{rm}}{\Longrightarrow}} (Sbba, abbSbba) \overset{(3)}{\underset{\text{rm}}{\Longrightarrow}} (bba, abbbba).$$

То, что в начале и в конце выходной цепочки должна быть порождена буква a, определяется лишь в момент, когда сканирование входной цепочки заканчивается и выясняется, что правилом (1) порождается буква a. Следовательно, вы-

Следовательно, выдача на выход может начаться только после того, как вся входная цепочка прочитана. Естественный способ получить на выходе цепочку w^R — запомнить w в магазине, а затем выдать цепочку w^R на выход, выбирая ее символы из магазина. Далее требуется на выходе сгенерировать цепочку w, но в магазине, пустом к этому времени, нет для этого никакой информации. Где еще, помимо магазина, могла бы быть информация для восстановления цепочки w? — Только в состояниях управления детерминированного магазинного преобразователя (dpdt). Но и там невозможно сохранить информацию о всей входной цепочке, так как она может быть сколь угодно большой длины. Короче говоря, dpdt, который мог бы реализовать описанную трансляцию, не существует.

Однако если простая синтаксически управляемая трансляция с входной грамматикой класса $LR(k)$ не требует, чтобы выходная цепочка порождалась до того, как устанавлено, какое правило применяется, то соответствующая трансляция может быть реализована посредством dpdt. Это приводит нас к понятию *постфиксной схемы синтаксически управляемой трансляции*.

***Определение* 3.16.** $T = (N, \Sigma, \Delta, R, \mathrm{S})$ называется *постфиксной схемой синтаксически управляемой трансляции*, если каждое ее правило имеет вид $A \to \alpha, \beta$, где $\beta \in N^*\Delta^*$.

Теорема 3.8. *Пусть* $T = (N, \Sigma, \Delta, R, S)$ *— простая семантически однозначная постфиксная схема синтаксически управляемой трансляции с входной* $LR(k)$*-грамматикой. Существует детерминированный магазинный преобразователь* P*, такой, что* $\tau(P) = \{(x\$, y) \mid (x, y) \in \tau(T)\}$.

Доказательство. По входной грамматике схемы T можно построить канонический $LR(k)$-анализатор, а затем моделировать его работу посредством dpdt P, накапливающего аванцепочку в состояниях и воспроизводящего действия shift и reduce i. При этом вместо выдачи на выходную ленту номера правила i он выдает выходные символы, входящие в состав семантической цепочки этого правила. В момент принятия входной цепочки dpdt P переходит в конечное состояние. Именно: если правило с номером i есть $A \to \alpha, \beta z$, где $\beta \in N^*$, $z \in \Delta^*$, то dpdt P выдает цепочку z на выход.

Технические детали построения dpdt P и доказательство его адекватности sdts T оставляем в качестве упражнения читателю.

Пример 3.14. Пусть имеется схема T с правилами 0) $S' \to S, S$; 1) $S \to SaSb, SSc$; 2) $S \to \varepsilon$, ε.

Входную грамматику этой схемы, являющуюся $LR(1)$-грамматикой во всех деталях мы обсуждали ранее. По ней была построена управляющая таблица адекватного канонического $LR(1)$-анализатора. Эта же таблица может быть использована $LR(1)$-транслятором, который отличается от анализатора только тем, что вместо номера правила пишет на выходную ленту выходные символы из семантической цепочки этого правила.

Пусть имеется следующий вывод в схеме:

$$(S, S) \underset{\text{rm}}{\overset{(1)}{\Rightarrow}} (SaSb, SSc) \underset{\text{rm}}{\overset{(1)}{\Rightarrow}} (SaSaSbb, SSScc) \underset{\text{rm}}{\overset{(2)}{\Rightarrow}} (SaSabb, SScc) \underset{\text{rm}}{\overset{(2)}{\Rightarrow}}$$
$$\underset{\text{rm}}{\overset{(2)}{\Rightarrow}} (Saabb, Scc) \underset{\text{rm}}{\overset{(2)}{\Rightarrow}} (aabb, cc).$$

Руководствуясь табл. 3.3, $LR(1)$-транслятор совершает следующие движения:

$$(T_0, aabb, \varepsilon) \underset{\text{g}}{\vdash} (T_0ST_1, aabb, \varepsilon) \underset{\text{g}}{\vdash} (T_0ST_1aT_2, abb, \varepsilon) \underset{\text{g}}{\vdash} (T_0ST_1aT_2ST_3, abb, \varepsilon) \underset{\text{g}}{\vdash}$$
$$\underset{\text{g}}{\vdash} (T_0ST_1aT_2ST_3aT_4, bb, \varepsilon) \underset{\text{g}}{\vdash} (T_0ST_1aT_2ST_3aT_4ST_6, bb, \varepsilon) \underset{\text{g}}{\vdash}$$
$$\underset{\text{g}}{\vdash} (T_0ST_1aT_2ST_3aT_4ST_6bT_7, b, \varepsilon) \underset{\text{g}}{\vdash} (T_0ST_1aT_2ST_3, b, c) \underset{\text{g}}{\vdash}$$
$$\underset{\text{g}}{\vdash} (T_0ST_1aT_2ST_3bT_5, \varepsilon, c) \underset{\text{g}}{\vdash} (T_0ST_1, \varepsilon, cc).$$

§ 3.9. Простые непостфиксные синтаксически управляемые *LR*-трансляции

Предположим, что имеется простая, но не постфиксная sdts, входная грамматика которой есть $LR(k)$-грамматика. Как реализовать такой перевод? Один из возможных методов состоит в использовании многопросмотровой схемы перевода на базе нескольких dpdt.

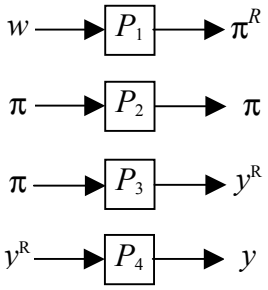

Рис. 3.3.

Пусть $T = (N, \Sigma, \Delta, R, S)$ — простая семантически однозначная sdts с входной $LR(k)$-грамматикой G. Для реализации трансляции, задаваемой схемой T, можно построить четырехуровневую схему перевода (рис. 3.3).

Первый уровень занимает dpdt P_1. Его входом служит входная цепочка w\$, а выходом π^R — правосторонний анализ цепочки w.

На втором уровне dpdt P_2 обращает цепочку π^R. Для этого ему достаточно поместить всю цепочку π^R в магазин типа last-in-first-out и прочитать ее из магазина, выдавая на выход. Получается цепочка π — последовательность номеров правил правостороннего вывода входной цепочки w. На следующем этапе она используется для порождения соответствующей инвертированной выходной цепочки правосторонним выводом в выходной грамматике схемы T.

Входом для третьего уровня служит цепочка π. Выход — перевод, определяемый простой sdts $T' = (N, \Sigma', \Delta, R', S)$, где R' содержит правило вида

$$A \to iB_mB_{m-1}\ldots B_1, y_mB_my_{m-1}B_{m-1}\ldots y_1B_1y_0$$

тогда и только тогда, когда $A \to x_0 B_1 x_1 \ldots B_m x_m$, $y_0 B_1\, y_1 \ldots B_m y_m$ — правило из R, а правило $A \to x_0 B_1\, x_1 \ldots B_m x_m$ есть правило номер i входной $LR(k)$-грамматики. Нетрудно доказать, что $(\pi, y^R) \in \tau(T')$ тогда и только тогда, когда $(S, S) \overset{\pi}{\underset{rm}{\Rightarrow}} (w, y)$.

Схема T' — это простая sdts, основанная на $LL(1)$-грамматике, и, следовательно, ее можно реализовать посредством dpdt P_3.

На четвертом уровне dpdt P_4 просто обращает цепочку y^R — выход P_3, записывая его в магазин типа first-in-last-out, а затем выдавая цепочку y из магазина на свой выход.

Число основных операций, выполняемых на каждом уровне, пропорционально длине цепочки w. Таким образом, можно сформулировать следующий результат:

Теорема 3.9. Трансляция, задаваемая простой семантически однозначной схемой синтаксически управляемой трансляции с входной LR(k)-грамматикой, может быть реализована за время, пропорциональное длине входной цепочки.

Доказательство представляет собой формализацию вышеизложенного.

§ 3.10. *LALR*(*k*)-Грамматики

На практике часто используются частные подклассы $LR(k)$-грамматик, анализаторы для которых имеют более компактные управляющие таблицы по сравнению с таблицами канонического $LR(k)$-анализатора. Здесь мы определим один из таких подклассов грамматик, называемых *LALR*(*k*)-*грамматиками*.

***Определение* 3.17.** *Ядром* $LR(k)$-ситуации $[A \to \beta_1.\beta_2,\ u]$ назовем $A \to \beta_1.\beta_2$. Определим функцию CORE($\mathcal{A}$), где $\mathcal{A}$ — некоторое множество $LR(k)$-ситуаций, как множество ядер, входящих в состав $LR(k)$-ситуаций из $\mathcal{A}$.

***Определение* 3.18.** Пусть G — контекстно-свободная грамматика, $\mathcal{S}$ — каноническая система множеств $LR(k)$-ситуаций для грамматики G и

$$\mathcal{S}' = \{\mathcal{A}' \mid \mathcal{A}' = \bigcup_{\mathcal{B} \in \mathcal{S}} \{\mathcal{B} \mid \mathrm{CORE}(\mathcal{B}) = \mathrm{CORE}(\mathcal{A})\},\ \mathcal{A} \in \mathcal{S}\}.$$

Если каждое множество $\mathcal{A}' \in \mathcal{S}'$ — непротиворечиво, то G называется *LALR*(*k*)-*грамматикой*.

Другими словами, если слить все множества $LR(k)$-ситуаций с одинаковыми наборами ядер в одно множество и окажется, что все полученные таким образом множества $LR(k)$-ситуаций непротиворечивы, то G — $LALR(k)$-грам-матика. Число множеств, полученных при слиянии, разве лишь уменьшится. Соответственно уменьшится число $LR(k)$-таблиц. Последние строятся обычным образом по объединенным множествам $LR(k)$-ситуаций. Очевидно, что корректность $LALR(k)$-анализатора, использующего таким образом полученные $LR(k)$-таблицы, не нуждается в доказательстве.

Пример 3.15. Проверим, является ли рассмотренная ранее $LR(1)$-грамматика с правилами 0) $S' \to S$, 1) $S \to SaSb$, 2) $S \to \varepsilon$ $LALR(1)$-грамматикой.

Каноническая система множеств $LR(1)$-ситуаций для нее была построена в примере 3.10:

$\mathscr{A}_0=\{[S'\to .S, \varepsilon], [S\to .SaSb, \varepsilon \mid a], [S\to ., \varepsilon \mid a]\}$;

$\mathscr{A}_1=\{[S'\to S., \varepsilon], [S\to S.aSb, \varepsilon \mid a]\}$;

$\mathscr{A}_2=\{[S\to Sa.Sb, \varepsilon \mid a], [S\to .SaSb, a \mid b], [S\to ., a \mid b]\}$;

$\mathscr{A}_3=\{[S\to SaS.b, \varepsilon \mid a], [S\to S.aSb, a \mid b]\}$;

$\mathscr{A}_4=\{[S\to Sa.Sb, a \mid b], [S\to .SaSb, a \mid b], [S\to ., a \mid b]\}$;

$\mathscr{A}_5=\{[S\to SaSb., \varepsilon \mid a\}$;

$\mathscr{A}_6=\{[S\to SaS.b, a \mid b], [S\to S.aSb, a \mid b]\}$;

$\mathscr{A}_7=\{[S\to SaSb., a \mid b]\}$.

Ясно, что в приведенной системе можно слить множества $\mathscr{A}_2$ и $\mathscr{A}_4$:

$$\mathscr{A}_{24}=\mathscr{A}_2\cup\mathscr{A}_4=\{[S\to Sa.Sb, \varepsilon \mid a \mid b], [S\to .SaSb, a \mid b], [S\to ., a \mid b]\},$$

множества $\mathscr{A}_3$ и $\mathscr{A}_6$:

$$\mathscr{A}_{36}=\mathscr{A}_3\cup\mathscr{A}_6=\{[S\to SaS.b, \varepsilon \mid a \mid b], [S\to S.aSb, a \mid b]\},$$

а также множества $\mathscr{A}_5$ и $\mathscr{A}_7$:

$$\mathscr{A}_{57}=\mathscr{A}_5\cup\mathscr{A}_7=\{[S\to SaSb., \varepsilon \mid a \mid b\}.$$

Полученные множества $\mathscr{A}_{24}$, $\mathscr{A}_{36}$ и $\mathscr{A}_{57}$ — не противоречивы. Системе объединенных множеств $LR(k)$-ситуаций соответствует управляющая таблица (табл. 3.6).

Табл. 3.6

$LR(1)$-таблицы	$f(u)$			$g(X)$		
	a	b	ε	S	a	b
T_0	reduce 2		reduce 2	T_1		
T_1	shift		accept		T_{24}	
T_{24}	reduce 2	reduce 2		T_{36}		
T_{36}	shift	shift			T_{24}	T_{57}
T_{57}	reduce 1	reduce 1	reduce 1			

Отметим, что анализатор, использующий $LALR(k)$-таблицы, может чуть запаздывать с обнаружением ошибки по отношению к анализатору, использующему каноническое множество $LR(k)$-таблиц. Например, канонический $LR(1)$-анализатор для рассматриваемой грамматики обнаруживает ошибку в цепочке abb, достигнув пятой конфигурации:

$$(T_0, abb, \varepsilon)\vdash(T_0ST_1, abb, 2)\vdash(T_0ST_1aT_2, bb, 2)\vdash(T_0ST_1aT_2ST_3, bb, 22)\vdash$$
$$\vdash(T_0ST_1aT_2ST_3bT_5, b, 22),$$

а $LALR(1)$-анализатор — на шестой:

$$(T_0, abb, \varepsilon)\vdash(T_0ST_1, abb, 2)\vdash(T_0ST_1aT_{24}, bb, 2)\vdash(T_0ST_1aT_{24}ST_{36}, bb, 22)\vdash$$
$$\vdash(T_0ST_1aT_{24}ST_{36}bT_{57}, b, 22)\vdash(T_0ST_1, b, 221).$$

Приложение

НЕРАЗРЕШИМЫЕ И РАЗРЕШИМЫЕ ПРОБЛЕМЫ, КАСАЮЩИЕСЯ ФОРМАЛЬНЫХ ЯЗЫКОВ

Здесь мы перечислим без доказательств алгоритмически неразрешимые и разрешимые проблемы, относящиеся к различным свойствам формальных язык.

П.1. Неразрешимые проблемы

Контекстно-зависимые языки. Проблема пустоты: дана контекстно-зависимая грамматика G. Вопрос: $L(G) = \varnothing$?

Контекстно-свободные языки.

Проблема пустоты пересечения: даны произвольные контекстно-свободные грамматики G_1 и G_2. Вопрос: $L(G_1) \cap L(G_2) = \varnothing$?

Проблема полноты: дана контекстно-свободная грамматика G, словарь терминалов которой есть Σ. Вопрос: $L(G) = \Sigma^*$?

Проблемы отношений между контекстно-свободными языками и регулярными множествами: даны контекстно-свободная грамматика G и регулярное множество R. Вопросы:

1) $L(G) = R$?

2) $L(G) \supseteq R$?

3) $\overline{L(G)} = \varnothing$?

Проблемы эквивалентности и вложенности: даны контекстно-свободные грамматики G_1 и G_2. Вопросы:

1) $L(G_1) = L(G_2)$?

2) $L(G_1) \subseteq L(G_2)$?

Проблема принадлежности пересечения классу контекстно-свободных языков: даны контекстно-свободные грамматики G_1 и G_2. Вопрос: $L(G_1) \cap L(G_2)$ — контекстно-свободный язык?

Проблема принадлежности дополнения классу контекстно-свободных языков: дана контекстно-свободная грамматика G. Вопрос: $\overline{L(G)}$ — контекстно-свободный язык?

Проблема регулярности языка: дана контекстно-свободная грамматика G. Вопрос: $L(G)$ — регулярное множество?

Неоднозначность контекстно-свободных грамматик и языков.

Неоднозначность контекстно-свободных грамматик: дана произвольная контекстно-свободная грамматика G. Вопрос: G — однозначна?

Проблема существенной синтаксической неоднозначности контекстно-свободных языков: дана произвольная контекстно-свободная грамматика G. Вопрос: $L(G)$ — существенно однозначен?

Детерминированные контекстно-свободные языки. Проблемы соотношения между детерминированными контекстно-свободными языками. Даны языки L_1 и L_2, принимаемые детерминированными магазинными автоматами. Вопросы:

1) $L_1 \cap L_2 = \varnothing$?

2) $L_1 \cap L_2$ — контекстно-свободный язык?

3) $L_1 \cup L_2$ — детерминированный?

4) $L_1 \subseteq L_2$?

П.2. Разрешимые проблемы,
касающиеся детерминированных
контекстно-свободных языков

Некоторые вопросы, которые не разрешимы для контекстно-свободных языков в общем случае, разрешимы для детерминированных языков. Например, даны детерминированный контекстно-свободный язык L и регулярное множество R. Разрешимы следующие вопросы:

1) L — существенно неоднозначен?

2) $L = R$?

3) L — регулярное множество?

4) $\overline{L} = \varnothing$?

5) $\overline{L}$ — контекстно-свободный язык?

6) $L \supseteq R$?

Нерешенная проблема — неизвестно, разрешима или нет следующая проблема: даны детерминированные магазинные автоматы M_1 и M_2. Вопрос: $T(M_1) = T(M_2)$?

УКАЗАТЕЛЬ ЛИТЕРАТУРЫ

1. **Greibach S.A.** A note on undecidable properties of formal languages // Math. Systems Theory. 1968. Vol. 2, №1. P.1–6.
2. **Hopcroft J.E., Ullman J.D.** Formal languages and their relation to automata. — Reading, MA: Addison-Wesley Pub. Co., Inc., 1969. 242 p.
3. **Rozenberg G., Salomaa A.** Handbook of Formal Languages. — Berlin, Heidelberg: Springer-Verlag, 1997. Vol.1: 873 p., Vol. 2: 528 p., Vol. 3: 625 p.
4. **Salomaa A.** Formal languages. — N.Y.: Academic Press, 1973. 335 p.
5. **Агафонов В.Н.** Синтаксический анализ языков программирования: Учеб. пособие. — Новосибирск: Изд-во НГУ, 1981. 91 с.
6. **Ахо А., Ульман Дж.** Теория синтаксического анализа, перевода и компиляции. Т. 1: Синтаксический анализ. 612 с.; Т.2: Компиляция. 487 с. — М.: Мир, 1978.
7. **Братчиков И.Л.** Синтаксис языков программирования. — М.: Наука, 1975. 232 с.
8. **Гинзбург С.** Математическая теория контекстно-свободных языков. — М.: Мир, 1970. 326 с.
9. **Гладкий А.В.** Формальные грамматики и языки. — М.: Наука, 1973. 368 с.
10. **Грис Д.** Конструирование компиляторов для цифровых вычислительных машин. — М.: Мир, 1975. 544 с.
11. **Гросс М., Лантен А.** Теория формальных грамматик. — М.: Мир, 1971. 294 с.
12. **Кнут Д.** Искусство программирования для ЭВМ. Т.1. — М.: Мир, 1976. 735 с.
13. **Льюис Ф., Розенкранц Д., Стирнз Р.** Теоретические основы проектирования компиляторов. — М.: Мир, 1979. 654 с.
14. **Рейуорд-Смит В. Дж.** Теория формальных языков: Вводный курс. — М.: Радио и связь, 1988. 129 с.
15. **Саломаа А.** Жемчужины теории формальных языков. — М.: Мир, 1986. 159 с.
16. **Семантика** языков программирования: Сб. статей под ред. А.Н.Маслова и Э.Д.Стоцкого — М.: Мир, 1980. 394 с.
17. **Фитиалов С.Я.** Формальные грамматики. — Л..: Изд-во Ленингр. ун-та, 1984. 99 с.
18. **Фостер Дж.** Автоматический синтаксический анализ. — М.: Мир, 1975. 71 с.
19. **Хантер Р.** Проектирование и конструирование компиляторов. — М.: Финансы и статистика, 1984. 232 с.
20. **Языки и автоматы:** Сб. статей под ред. В.М.Курочкина. — М.: Мир, 1975. 358 с.

ИБН
Научная литература

Борис Константинович Мартыненко

Языки и трансляции
Учебное пособие

Редактор *Т.В. Мызникова*
Компьютерная верстка *Б.К. Мартыненко*