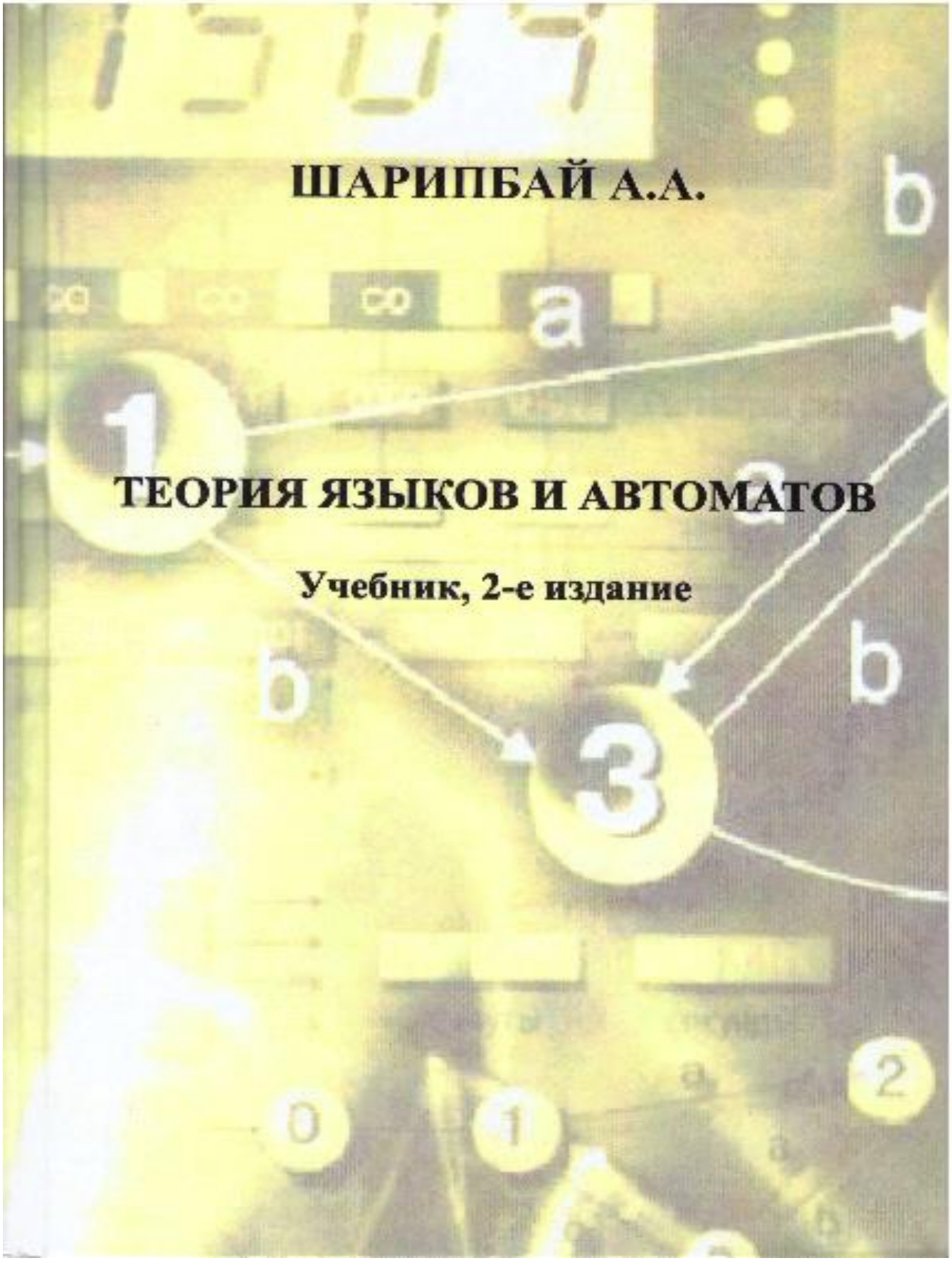


ШАРИПБАЙ А.А.

ТЕОРИЯ ЯЗЫКОВ И АВТОМАТОВ

Учебник, 2-е издание

ШАРИПБАЙ А.А.

ТЕОРИЯ ЯЗЫКОВ И АВТОМАТОВ

Учебник

(Издание второе переработанное и дополненное)

УДК 004.4(075.8)
ББК 32.973.26-018.1я73
Ш-25

Рецезенты:

Адамов А.А. – заведующий кафедрой Математического и компьютерного моделирования, ЕНУ имени Л.Н.Гумилева, д.т.н., профессор;

Казиев Г.З. – исследовательского центра АО «Зерде холдинг», д.т.н., профессор;

Тукеев У.А. – профессор кафедры Информационных систем КазНУ имени Аль-Фараби, д.т.н., профессор.

Автор:

Ш-25 Шарипбай А.А – директор НИИ «Искусственный интеллект», профессор кафедры «Теоретическая информатика» ЕНУ имени Л.Н. Гумилева, доктор технических наук, профессор, лауреат Государственной премии Республики Казахстан, академик Международной академии информатизации, академик Академии педагогических наук Казахстана.

Одобрено Ученым советом Евразийского национального университета имени Л.Н. Гумилева

Теория языков и автоматов: учебник/ авт. Шарипбай А.А.– Алматы: издательство Эверо, 2015. –224 с.

В учебнике рассматриваются различные типы механизмов порождения и распознавания языков, их структуры, принципы работы и свойства, показываются многочисленные примеры, даются задания, ставятся вопросы и предлагаются тесты.

Содержание учебника составлено в соответствии с государственным общеобязательным стандартом образования по специальности Информатика и с целью применить кредитную технологию и имеет три уровня, что позволит проводить на четырех уровнях (текущий, промежуточный, рубежный и итоговый) контроль знаний:

Учебником могут пользоваться студенты, магистранты, докторанты и преподаватели специальностей Вычислительная техника и программное обеспечение, Информационные системы, Автоматизация и управление, а также ученые и другие, занимающиеся исследованием или приложением теории языков и автоматов.

УДК 004.4(075.8) ББК 32.973.26-018.1я73
ISBN 978 -601-7454-77-7

Оглавление

ПРЕДИСЛОВИЕ К 1-МУ ИЗДАНИЮ **6**

ПРЕДИСЛОВИЕ К 2-МУ ИЗДАНИЮ **10**

I. ОСНОВЫ ЯЗЫКОВ И АВТОМАТОВ **11**

I.1. Обозначения, понятия и сокращения **11**
I.1.1. Обозначения 11
I.1.2. Понятия 14
I.1.3. Сокращения 15

I.2. Математические основы **18**
I.2.1. Множества и операции над ними 18
I.2.2. Отношения и способы их представления 24
I.2.3. Символы, цепочки и операции над ними 33
I.2.4. Алфавит, слово, язык и операции над языками 39

I.3. Механизмы определения языков **46**
I.3.1. Порождающие механизмы языков 46
I.3.3. Механизмы распознавания языков 58
I.3.4. Типы автоматов и их алгоритмические проблемы 62
I.3.5. Формальное определение автомата 67

II. РЕГУЛЯРНЫЕ ЯЗЫКИ **71**

II.1. Механизмы порождения регулярных языков **71**
II.1.1. Регулярные множества и регулярные выражения 71
II.1.2. Регулярная алгебра и регулярные уравнения 77
II.1.3. Системы регулярных уравнений 83
II.1.4. Линейные грамматики 90

II.2. Распознающие механизмы регулярных языков **94**
II.2.1. Недетерминированные и детерминированные конечные автоматы 94 94
II.2.2. Эквивалентность недетерминированых и детерминированных конечных автоматов 105
II.3.2. Эквивалентность регулярных выражений и конечных автоматов 112 112
II.3.3. Эквивалентность праволинейных грамматик и конечных автоматов 122 122

III. БЕСКОНТЕКСТНЫЕ ЯЗЫКИ **131**

III.1. Порождающие механизмы бесконтекстных языков **131**
III.1.1. Бесконтекстные грамматики 131
III.1.2. Преобразование безконтекстных грамматик 137
III.1.3. Нормальная форма Хомского 146
III.1.4. Нормальная форма Грейбах 148

III.2. Распознающие механизмы бесконтекстных языков **153**
III.2.1. Состав и структура стековых автоматов 153
III.2.2. Цепочки и языки, распознаваемые стековыми автоматами 160

III.3. Свойства бесконтекстных языков **171**
III.3.1. Эквивалентность беконтекстных грамматик и 171
стековых автоматов 171
III.3.2. Алгоритмические проблемы бесконтекстных грамматик 177

IV. КОНТЕКСТНЫЕ ЯЗЫКИ **180**

IV.1. Порождающие механизмы контекстных языков **180**
IV.1.1. Контекстные грамматики 180

IV.2. Распознающие механизмы контекстных языков **187**
IV.2.1. Линейно-ограниченные автоматы 187

IV.3. Свойства контекстных языков **192**
IV.3.1. Эквивалентность контекстных грамматик и 192
линейно-ограниченных автоматов 192
IV.3.2. Алгоритмические проблемы контекстных языков 197

V. НЕОГРАНИЧЕННЫЕ ЯЗЫКИ **200**

V.1. Порождающие механизмы неограниченных языков **200**
V.1.1. Неограниченные грамматики 200

V.2. Распознающие механизмы неограниченных языков **202**
V.2.1. Машины Тьюринга 202

V.3. Свойства неограниченных языков **210**
V.3.1. Эквивалентность неограниченных грамматик и 210
машин Тьюринга 210
V.3.2. Алгоритмические проблемы неограниченных языков 217

ЛИТЕРАТУРНЫЕ ИСТОЧНИКИ И ИНТЕРНЕТ РЕСУРСЫ **220**

ПРЕДИСЛОВИЕ К ПЕРВОМУ ИЗДАНИЮ

В эпоху информатизации общества *Информатика - Computer Science* (*Вычислительная наука*) - стала самой нужной и самой важной наукой. С ней связывают революцию в области накопления, обработки и передачи информации, которая следует за революциями в овладении веществом и энергией, затрагивает и коренным образом преобразует не только сферу материального производства, но и интеллектуальную, духовную сферы жизни человеческого общества.

Информатика занимается проблемами создания и исследования информационных технологий, которые реализуются на компьютере с помощью программ, написанных на искусственных языках, для решения различных задач и автоматизации отдельных процессов в интеллектуальной жизни человека.

Информатика состоит из *Технического обеспечения – Hardware* (*Твердые средства*), *Программного обеспечения – Software* (*Мягкие средства*) и *Интеллектуального обеспечения – Brainware* (*Умные средства*). Эти части равнозначны и способны модифицироваться, приспосабливаясь к современным требованиям развития человеческого общества.

В информационных технологиях лингвистические (морфологические, синтаксические, семантические и речевые) методы анализа и синтеза играют важную роль. Различные языковые процессоры (трансляторы, компиляторы, интерпретаторы, генераторы, конверторы, редакторы и др.) основаны на применении лингвистических методов. Теория языков и автоматов, являясь фундаментальным направлением науки информатики, составляет научную основу этих методов и занимается исследованием порождающих и распознающих механизмов языков.

Основы теории порождающих механизмов (формальных грамматик) были заложены Н. Хомским в 40–50-е годы XX века в связи с его лингвистическими работами, посвященными изучению естественных языков [23]. А основы теории распознающих механизмов

(конечных автоматов) были созданы в те же годы M.O.Rabin, D.Scott [27] и В.М.Глушковым [10].

В 60-е того же столетия теория языков и автоматов нашли широкое практическое применение в области разработки и реализации языков программирования. В настоящее время применение технологических средств, основанных на лингвистические методы, существенно расширились. Они применяются при создании систем обработки естественных языков, в том числе орфографических корректоров, систем смыслового поиска и принятий решений, а также в речевых технологиях. Однако их грамотное и эффективное применение требует от пользователя знания, по крайней мере, основ математической теории, на которой они базируются.

Предлагаемый учебник написан на основе лекций, читавшихся автором в разные годы для студентов и магистрантов факультета информационных технологий Евразийского национального университета имени Л.Н. Гумилева по специальностям Информатика, Информационные системы, Вычислительная техника и программное обеспечение.

Причиной подготовки данного учебника послужила отсутствие учебных изданий, содержания которых полностью соответствовали бы государственному общеобязательному стандарту образования. Поэтому для полноценного обеспечения учебного процесса автору приходилось использовать различные литературы и интернет ресурсы, в которых одни и те же понятия имели разные термины, определения и обозначения. Это обстоятельство и натолкнуло на идею написания данного учебника с использованием единых терминов, определений и обозначений для одних и тех же понятий. Автор надеется, что предлагаемый учебник поправит существующую нехватку учебных изданий по изучаемому предмету и позволит унифицировать терминов, определений и обозначений одних и тех же понятий.

Методика изложения учебного материала сводится к тому, чтобы рассказать читателю о фундаментальных фактах классической теории языков и автоматов, познакомить его с приемами доказательства утверждений в изучаемой области, а также дать ему некоторый запас

примеров. Почти все доказательства носят конструктивный характер и потому дают полезный материал для практики.

Учебник одновременно написан на русском и казахском языках с одинаковым содержанием и состоит из трех уровненных пяти частей и списка литературы.

В первой части учебника даются используемые обозначения, понятия, пояснения, сокращения, математические основы и механизмы определения языков.

Во второй части учебника рассматриваются механизмы порождения (регулярные множества, регулярные выражения, регулярная алгебра, регулярные уравнения и их системы, праволинейная грамматика), механизмы распознавания языков (недетерминированные и детерминированные конечные автоматы и их эквивалентность), свойства регулярных языков (эквивалентность регулярных выражений, праволинейных грамматик и конечных автоматов) и их алгоритмические проблемы. Надо заметить, что регулярные языки образуют класс языков, занимающий центральное положение в теории формальных языков.

В третьей части учебника описываются механизмы порождения бесконтекстных языков (контекстно-свободные грамматики), механизмы распознавания бесконтекстных языков (недетерминированные и детерминированные стековые автоматы), свойства бесконтекстных языков (эквивалентность контекстно-свободных грамматик и стековых автоматов) и их алгоритмические проблемы.

В четвертой части учебника описываются механизмы порождения контекстных языков (контекстно-зависимые грамматики), механизмы распознавания контекстных языков (недетерминированные и детерминированные линейно-ограниченные автоматы), свойства контекстных языков (эквивалентность контекстно-зависимых грамматик и линейно-ограниченных автоматов) и их алгоритмические проблемы.

В пятой части учебника предлагаются механизмы порождения неограниченных языков (неограниченные грамматики), механизмы

распознавания неограниченных языков (недетерминированные и детерминированные машины Тьюринга), свойства неограниченных языков (эквивалентность неограниченных грамматик и машин Тьюринга) и их алгоритмические проблемы.

В списке литературы приводятся общие литературные источники, которые были использованы при формировании учебного материала учебника. Среди них имеются широко распространенные монографии, учебники, учебные пособия и др. Поэтому в тексте настоящего учебника не указаны прямые ссылки на них.

Учебник предназначен студентам, магистрантам, докторантам, преподавателям, ученым и всем тем, кто самостоятельно желает изучить теорию языков и автоматов.

Автор искренно благодарит рецензентов Адамова А.А, Казиева У.А. и Тукеева У.А. за ценные замечения и предложения, а также приносит свою признательность Сагадиевой А. за техническую редакцию учебника.

Автор:
Алтынбек Амирович ШАРИПБАЙ,
доктор технических наук, профессор,
лауреат Государственной премии Республики Казахстан.

ПРЕДИСЛОВИЕ К ВТОРОМУ ИЗДАНИЮ

Основой для переиздания настоящего учебника послужила его актуальность, поскольку спрос на него не утихает, а возрастает. Читатели могут найти этот учебник в библиотеках, но они не могут приобрести его в магазинах из-за ограниченности тиража 1-го издания. Это обстоятельство и побудило автора на переиздание этого учебника.

При нынешнем переиздании настоящего учебника моим первоначальным намерением было сохранить в нетронутом виде упомянутое выше предисловие к 1-му изданию. Но несмотря на это я добавлю, что в учебнике будут рассмотрены фундаментальные и прикладные проблемы языков и автоматов, необходимые для разработки современных языковых процессоров (конвертеров, верификаторов, редакторов, трансляторов, компиляторов, интерпретаторов и другие).

В начале каждого параграфа поставленные ссылки на литературные источники и интернет ресурсы, которые были использованы при формировании учебного материала.

Учебник в первую очередь рекомендуется студентам, магистрантам и докторантам различных специальностей направления "Информационные и коммуникационные технологии" (ИКТ). Кроме того, он может быть полезен ИКТ-специалистам (аналитикам, программистам), преподавателям и ученым. И наконец, учебник может заинтересовать всех, кто имеет дело с формальными языками и автоматами и хочет больше узнать о технологии создания современных языковых процессоров.

Автор благодарит своих рецензентов за ценные замечания и предложения, которые позволили существенно улучшить первоначальную версию 2-го издания настоящего учебника.

Все замечания и предложения, касающиеся учебнику, можно выслать на следующий e-mail адрес: sharalt@mail.ru.

Автор:

Алтынбек Амирович ШАРИПБАЙ,

I. ОСНОВЫ ЯЗЫКОВ И АВТОМАТОВ

I.1. Обозначения, понятия и сокращения

I.1.1. Обозначения

В этом параграфе даются используемые обозначения, их названия и применения. Они показаны в таблице I.1.1.

Таблица I.1.1. Обозначения.

№	Обозначения	Названия и применения
1.	=	Равенство: показывает, что величина в левой части равна величине в правой части
2.	≠	Неравенство: показывает, что величина в левой части не равна величине в правой части
3.	∈	Принадлежность: показывает, что величина в левой части принадлежит величине в правой части
4.	∉	Непринадлежность: показывает, что величина в левой части не принадлежит величине в правой части
5.	∅	Пустое множество: показывает, что в множестве нет ни одного элемента
6.	ε	Пустая цепочка: показывает, что в цепочке нет ни одного символа
7.	{ε}	Множество пустых цепочек: элементом множества является толька одна пустая цепочка
8.	␣	Пробел: знак, служащий разделителем между цепочками и имеет код в таблицах кодировки, но не виден при выводе на экран и печать.
9.	⊆	Вхождение: показывает, что величина в левой части входит в величину в правой части
10.	⊇	Содержание: показывает, что величина в левой части содержит величину в правой части
11.	⊴	Префикс: показывает, что цепочка в левой части является начально подцепочки в правой части
12.	⊵	Постфикс: показывает, что цепочка в левой части является конечной поцепочки в правой части
13.	***U***	Универсальное множество (универсум): показывает, что множество содержит все элементы определенного типа

14.	∩	Операция пересечения: показывает, что величина в левой части пересекается с величиной в правой части
15.	∪	Операция объединение: показывает, что величина в левой части объединяется с величиной в правой части
16.	\	Операция разности: показывает, что от величины в левой части отнимается величина в правой части
17.	×	Операция прямого - декартово произведения: показывает прямое произведение величина в левой части и величина в правой части
18.	·	Операция сцепления - конкатенация: показывает сцепление двух или более величинн
19.	∨	Операция выбора - дизъюнкция: показывает альтернативу двух или более величин
20.	*	Операция итерация: показывает объединение (альтернативу) кратных сцеплений величин на себя
21.	+	Операция положительная итерация: показывает объединение (альтернативу) кратных сцеплений непустых величин на себя
22.	< , >	Представление имен понятий: показывает, что имена понятий находятся внутри угловой скобки
23.	{, }	Фигурная скобка: показывает, что обязательные элементы находятся внутри фигурной скобки
24.	**[,]**	Квадратная скобка: показывает, что необязательные элементы находятся внутри квадратной скобки
25.	→	Разделитель правила: показывает, что в левой части находится определяемый элемент, а в правой части – определяющие элементы
26.	⇒	Отношение вывода: показывает отношение между цепочкой в правой части и цепочки в левой части, полученное применением правил грамматики
27.	⊥	Указатель дна стека: показывает, что в стеке ничего нет
28.	⊢,⊣	Указатели начала и конца: показывает левого и правого края ленты

29.	$\models$	Отношение между конфигурациями автомата
30.	$\boldsymbol{L(G)}$	Язык, порождаемый грамматикой $\boldsymbol{G}$
31.	$\boldsymbol{L(M)}$	Язык, распознаваемый автоматом $\boldsymbol{M}$
32.	$\rightleftharpoons$	По определению: объект находящийся в левой части по определению равен выражению в правой части

Примеры I.1.1:

1. $\boldsymbol{A}=\boldsymbol{B}$ означает, что величина $\boldsymbol{A}$ и величина $\boldsymbol{B}$ равны.
2. $a \in \boldsymbol{A}$ показывает принадлежность величины a к величине $\boldsymbol{A}$.
3. $\boldsymbol{A} \subseteq \boldsymbol{B}$ означает, что величина $\boldsymbol{A}$ входит в величину $\boldsymbol{B}$.
4. $\boldsymbol{A} \cup \boldsymbol{B}$ означает, что величина $\boldsymbol{A}$ и величина $\boldsymbol{B}$ объединяются.
5. $\boldsymbol{A} \cdot \boldsymbol{B}$ означает, что величина $\boldsymbol{A}$ и величина $\boldsymbol{B}$ сцепляются.
6. $\boldsymbol{A} \cap \boldsymbol{B}$ означает, что величина $\boldsymbol{A}$ и величина $\boldsymbol{B}$ пересекаются.
7. $\boldsymbol{A} \rightleftharpoons \boldsymbol{a}+\boldsymbol{b}$ означает, что величина $\boldsymbol{A}$ по определению равна $\boldsymbol{a}+\boldsymbol{b}$.

З**адания I.1.1**:

1. Записать неравенство величины $\boldsymbol{A}$ и величины $\boldsymbol{B}$.
2. Записать непринадлежность величины a к величине $\boldsymbol{A}$.
3. Записать содержимость величины $\boldsymbol{B}$ в величине $\boldsymbol{A}$.
4. Записать пересечение величины $\boldsymbol{A}$ и величины $\boldsymbol{B}$.
5. Записать разность величины $\boldsymbol{A}$ и величины $\boldsymbol{B}$.
6. Записать прямое произведение величины $\boldsymbol{A}$ и величины $\boldsymbol{B}$.
7. Записать вхождение величины $\boldsymbol{A}$ в величину $\boldsymbol{B}$.

Вопросы I.1.1:

1. Как обозначается пустое множество?
2. Как обозначается множество пустых цепочек?
3. Как обозначается универсум?
4. Как показываются обязательные элементы?
5. Как показываются начало и конец ленты?
6. Как обозначается отношение вывода?
7. Как обозначается операция выбора?

I.1.2. Понятия

В этом параграфе приводятся используемые понятия и их пояснения. Они показаны в таблице I.1.2.

Таблица I.1.2. Понятия.

№	Понятия	Пояснения
1.	*Абстрактный автомат*	Алгоритмическая система, распознающая или преобразующая язык
2.	*Алфавит*	Непустое множество символов для обозначения чего-либо.
3.	*Длина цепочки*	Функция, показывающая количество символов в цепочке
4.	*Лингвистика*	Наука, которая исследует группу языков, сходство и разницу между ними
5.	*Непустая цепочка*	Цепочка, состоящая из одного или нескольких символов алфавита
6.	*Нетерминал*	Переменная величина, определяемая через другие постоянные или переменные величины
7.	*Пробел*	Символ, который используется для разделения цепочек друг от друга
8.	*Пустая цепочка*	Цепочка, которая не содержит ни одного символа алфавита и имеет длину нуль
9.	*Пустое множество*	Множество, не содержащее ни одного элемента
10.	*Распознаватель*	Абстрактный автомат, распознающий некоторый язык - множество цепочек в заданном алфавите
11.	*Семантика*	Описание соответствия между текстом и смыслом (значением)
12.	*Семиотика*	Наука о символьной системе со смыслом
13.	*Символ*	Минимальная неделимая синтаксическая единица
14.	*Синтаксис*	Набор правил, определяющий множество текстов со смыслом независимо от цели и обязанности языка
15.	*Текст*	Цепочка символов
16.	*Терминал*	Постоянная самоопределенная величина

		(буква, цифра, специальный знак)
17.	*Формальная грамматика*	Математическая система, порождающая формальный язык
18.	*Цепочка*	Конечная последовательность символов в заданном алфавите
19.	Язык	Множество конечных цепочек с определенными структурами и значениями

Примеры I.1.2:

1. Множество из 0 и 1 является алфавитом машинного языка.
2. Классический латинский алфавит состоит из 26 букв.
3. Не все буквы алфавита русского языка обозначают звуки.
4. Нетерминалами являются понятие «Предложение», «Слово».
5. Терминалами являются 0,1,3,4,5,6,7,8,9.

Задания I.1.2:

1. Дайте понятие цепочки.
2. Дайте понятие лингвистики.
3. Дайте понятие нетерминала.
4. Дайте понятие распознавателя.
5. Дайте понятие формальной грамматики.
6. Дайте понятие алфавит.
7. Дайте понятие абстрактного автомата.
8. Дайте понятие пустого множества.

Вопросы I.1.2:

1. Что такое длина цепочки?
2. Что такое пустая цепочка?
3. Что такое пустое множество?
4. Что такое пробел?
5. Что такое язык?
6. Что такое формальная грамматика?
7. Что такое распознаватель?
8. Что такое символ?

I.1.3. Сокращения

В таблице I.1.3 приводятся сокращения и их раскрытия.

Таблица I.1.3. Сокращения.

№	Сокращения	Раскрытия
1.	*БГ*	Бесконтектсная грамматика
2.	*ДКА*	Детерминированный конечный автомат
3.	*ДЛОА*	Детерминированный линейно ограниченный автомат
4.	*ДМТ*	Детерминированная машина Тьюринга
5.	*ДСА*	Детерминированный стековый автомат
6.	*КА*	Конечный автомат
7.	*КГ*	Контекстная грамматика
8.	*КЗГ*	Контекстно-зависимая грамматика
9.	*КСГ*	Контекстно-свободная грамматика
10.	*ЛЛГ*	Леволинейная грамматика
11.	*ЛОА*	Линейно-ограниченный автомат
12.	*МТ*	Машина Тьюринг
13.	*НГ*	Неограниченная грамматика
14.	*НКА*	Недетерминированный конечный автомат
15.	*НЛОА*	Недетерминированный линейно ограниченный автомат
16.	*НМТ*	Недетерминированная машина Тьюринга
17.	*НСА*	Недетерминированный стековый автомат
18.	*ПЛГ*	Праволинейная грамматика
19.	*РВ*	Регулярное выражение
20.	*РМ*	Регулярное множество
21.	*СА*	Стековый автомат

Примеры I.1.3:

1. ФГ – формальная грамматика.
2. КЛ – компьютерная лингвистика.
3. МЛ – математическая лингвистика.

4. ТЯА – теория языков и автоматов.
5. РА – регулярная алгебра.
6. КСГ – контекстно-свободная грамматика.
7. МТ – машина Тьюринга.
8. НКА – недетерминированный конечный автомат.
9. ПЛГ – праволинейная грамматика.
10. НГ – неограниченная грамматтика.
11. НЛОА – Недетерминированный линейно ограниченный автомат.

Задания I.1.3:

1. Сократите недетерминированный конечный автомат.
2. Сократите детерминированная машина Тьюринга.
3. Сократите леволинейная грамматика.
4. Сократите линейно-ограниченный автомат.
5. Сократите неограниченная грамматика.
6. Сократите стековый автомат.
7. Сократите недетерминированная машина Тьюринга.
8. Сократите бесконтекстная грамматика.
9. Сократите линейно-ограниченный автомат.
10. Сократите праволинейная грамматика.

Вопросы I.1.3:

1. Что означает ДЛОА?
2. Что означает БКГ?
3. Что означает ДКА?
4. Что означает НЛОА?
5. Что означает КСГ?
6. Что означает РВ?
7. Что означает НМТ?
8. Что означает РА?
9. Что означает РМ?
10. Что означает НДКА?

I.2. Математические основы

I.2.1. Множества и операции над ними

В этом параграфе нами будут рассмотрены понятие множества, определены операции над ними и раскрыты их свойства, а также предложены примеры, даны задания, сформулированы вопросы [14,17,22,27,28,33].

Определение I.2.1.1. *Множеством* называется объединение отдельных (дискретных) элементов, выбранных по некоторому признаку (критерию, типу).

Множество задается двумя способами: *перечислением всех элементов* или *описанием свойств элементов*. При этом его элементы указываются в фигурных скобках { и }.

Множества обозначаются прописными (заглавными) латинскими буквами с индексами или без них, а их элементы − строчными (малыми) латинскими буквами или арабскими цифрами. Например, множество натуральных чисел обозначается $\boldsymbol{N}$, а его элементы цифрами – 1,2,3,… .

Запись $a \in \boldsymbol{A}$ ($a \notin \boldsymbol{A}$) означает, что элемент a принадлежит (не принадлежит) множеству $\boldsymbol{A}$.

Примеры I.2.1.1:

1. Все неотрицательные целые числа образуют множество натуральных чисел.

2. $\boldsymbol{D} = \{0, 1\}$, где элементами множества $\boldsymbol{D}$ являются только перечисленные постоянные величины 0, 1.

3. $\boldsymbol{X} = \{x: \ x>0\}$, где элементами множества $\boldsymbol{X}$ являются только положительные переменные величины x.

Среди всех множеств выделяют два особых множества:

1. $\emptyset$ *– пустое множество*, не содержащее ни одного элемента.

2. $\boldsymbol{U}$ *– универсальное множество* (*универсум*), содержащее все элементы рассматриваемого типа (предметной области).

Относительно теории универсум это множество, содержащее в качестве элементов все объекты, рассматриваемые в этой теории.

Например, универсумом является:

1) в теории чисел – множество всех целых чисел;

2) в теории языков – множество всех слов в заданном алфавите;

3) в геометрии – множество всех точек *n*-мерного геометрического пространства.

Определение I.2.1.2. Мощность множества A равна числу его элементов и обозначается через $|A|$.

Примеры I.2.1.2:

1. $|\varnothing| = 0$.

2. Если $A = \{a, b, c, d, e\}$, то $|A| = 5$.

Определения I.2.1.3. Пусть заданы два множества A и B, тогда над ними можно определить следующие операции:

1. *Объединение* множеств A и B состоит из элементов A или B:

$$A \cup B = \{x: \ x \in A \vee x \in B\}.$$

2. *Пересечение* множеств A и B состоит из элементов A и B:

$$A \cap B = \{x: \ x \in A \ \& \ x \in B\}.$$

3. *Дополнение* к множеству A состоит из элементов универсума U и не включает элементов A:

$$\overline{A} = \{x| \quad x \in U \ \& \ x \notin A\}.$$

4. *Разность* множеств A и B состоит из элементов множества A и не включает элементов B:

$$A \setminus B = \{x: \ x \in A \ \ \& \ x \notin B\}.$$

5. *Симметрическая разность* множеств A и B состоит только из элементов A или только из элементов B:

$$A \triangle B = \{x: \ (x \in A \ \& \ x \notin B) \vee (x \in B \ \& \ x \notin A)\}$$

6. *Декартовое* (*прямое*) *произведение* множеств A и B состоит из всевозможных упорядоченных пар элементов A и B:

$$A \times B = \{(a, b): \ a \in A \ \& \ b \in B\}$$

Операции (1) – (3) могут быть представлены с помощью диаграммы Эйлера-Венна (Рис. *I*.2.1), в которой универсум U изображается прямоугольником, а множества A и B –окружностями. Для выделения результата применяется штриховка.

Здесь показано, что множества $\boldsymbol{A}$ и $\boldsymbol{B}$ являются подмножествами $\boldsymbol{U}$, и они записываются как $\boldsymbol{A} \subseteq \boldsymbol{U}$ и $\boldsymbol{B} \subseteq \boldsymbol{U}$ (см. *I*.2.2.).

Операции (1) – (3) можно определить не только над двумя множествами, но и над n множествами $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n$, где $n \in \boldsymbol{N}$ & $n>2$.

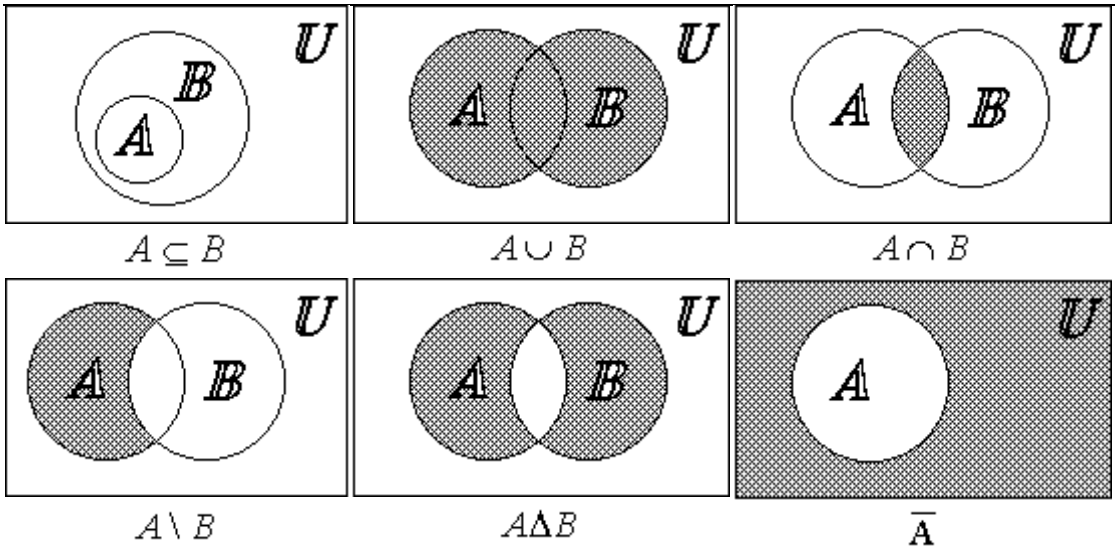


Рис. *I*.2.1. Диаграммы Эйлера-Венна.

Объединение над множествами $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n$ определяется как:

$$\boldsymbol{A}_1 \cup \boldsymbol{A}_2 \cup \ldots \cup \boldsymbol{A}_n = \bigcup_{i=1}^{n} \boldsymbol{A}_i .$$

Пересечение над множествами $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n$ определяется как:

$$\boldsymbol{A}_1 \cap \boldsymbol{A}_2 \cap \ldots \cap \boldsymbol{A}_n = \bigcap_{i=1}^{n} \boldsymbol{A}_i .$$

Прямое произведение над множествами $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_n$ определяется как множество кортежей вида $(a_1, a_2, \ldots, a_n)$, $a_1 \in \boldsymbol{A}_1, a_2 \in \boldsymbol{A}_2, \ldots, a_n \in \boldsymbol{A}_n$, т.е.

$$\boldsymbol{A}_1 \times \boldsymbol{A}_2 \times \ldots \times \boldsymbol{A}_n = \{(a_1, a_2, \ldots, a_n):\ a_1 \in \boldsymbol{A}_1 \,\&\, a_2 \in \boldsymbol{A}_2 \,\&\ldots\&\, a_n \in \boldsymbol{A}_n\}.$$

Здесь, если $\boldsymbol{A}_1 = \boldsymbol{A}_2 = \ldots = \boldsymbol{A}_n = \boldsymbol{A}$, то $\underbrace{\boldsymbol{A} \times \boldsymbol{A} \times \ldots \times \boldsymbol{A}}_{n} = \boldsymbol{A}^{n}$ – степень.

Примеры I.2.1.3. Пусть $\boldsymbol{A} = \{a, b, c, d, e, f\}$, $\boldsymbol{B} = \{c, d\}$, то:

1. $\boldsymbol{A} \cup \boldsymbol{B} = \{a, b, c, d, e, f\}$;
2. $\boldsymbol{A} \cap \boldsymbol{B} = \{c, d\}$;
3. $\boldsymbol{B} \times \boldsymbol{B} = \{(c, c), (c, d), (d, c), (d, d)\}$;
4. $\boldsymbol{A} \setminus \boldsymbol{B} = \{a, b, e, f\}$;
5. $\boldsymbol{A} \triangle \boldsymbol{B} = \{a, b, e, f\}$
6. $\overline{A}$ зависит от того, какой будет универсум $\boldsymbol{U}$. Допустим, если $\boldsymbol{U} = \{a, b, c, d, e, f, h\}$, то $\overline{A} = \{h\}$.

Теперь можно показать табличный способ задания множеств и операции над ними. Пусть заданы U, $A \subseteq U$ и $x \in U$.

Определение I.2.1.4. *Индикаторной* (*характеристической*) функцией для множества A называется функция $I_A(x)$, заданная как:

$$I_A(x)=\begin{cases}1, & \textit{если} \quad x \in A \\ 0, & \textit{если} \quad x \notin A\end{cases}$$

Таким образом: $I_A: U \to \{0,1\}$.

Для $A \subseteq U$ и $B \subseteq U$ имеют место следующие свойства:

$$I_A(x)=I_B(x) \Leftrightarrow A=B;$$

$$I_A(x) \le I_B(x) \Leftrightarrow A \subseteq B;$$

$$I_{\bar{A}}(x)=1-I_A(x);$$

$$I_{A \cup B}(x)=I_A(x)+I_B(x)-I_A(x) \cdot I_B(x);$$

$$I_{A \cap B}(x)=I_A(x) \cdot I_B(x);$$

$$I_{A \setminus B}(x)=I_A(x)-I_A(x) \cdot I_B(x);$$

$$I_{A \Delta B}(x)=I_A(x)+I_B(x)-2I_A(x) \cdot I_B(x).$$

Индикаторы удобно задавать с помощью таблицы I.2.1.

Таблица I.2.1. Индикаторы.

$x \in A$	$x \in B$	$x \in A \cup B$	$x \in A \cap B$	$x \in A \setminus B$	$x \notin A$	$x \in A \Delta B$
0	0	0	0	0	1	0
0	1	1	0	0	1	1
1	0	1	0	1	0	1
1	1	1	1	0	0	0

Операции над множествами обладают следующими свойствами:

I. Объединение, пересечение и разность:

1) $A \cup \varnothing = A$ – свойство нуля;
2) $A \cup A = A$ – идемпотентность;
3) $A \cup B = B$, если все элементы A содержатся в B;

4) $A \cup B = B \cup A$ – коммутативность;

5) $(A \cup B) \cup C = A \cup (B \cup C) = A \cup B \cup C$ – ассоциативность;

6) $A \cap \varnothing = A$ – свойство нуля;

7) $A \cap A = A$ – идемпотентность;

8) $A \cap B = A$, если A все элементы A содержатся в B;

9) $A \cap B = B \cap A$– коммутативность;

10) $(A \cap B) \cap C = A \cap (B \cap C) = A \cap B \cap C$ – ассоциативность;

11) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ – дистрибутивность;

12) $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ – дистрибутивность;

13) $A \cap (B \setminus C) = (A \cap B) \setminus (A \cap C)$ – дистрибутивность;

14) $A \cup \overline{A} = U$ – свойство дополнения;

15) $A \cap \overline{A} = \varnothing$ – свойство дополнения;

16) $\overline{A \cup B} = \overline{A} \cap \overline{B}$ – закон де Моргана;

17) $\overline{A \cap B} = \overline{A} \cup \overline{B}$ – закон де Моргана;

18) $\overline{\overline{A}} = A$ – инволютивность;

19) $A \setminus \varnothing = A$ – свойство разности;

20) $A \setminus A = \varnothing$ – свойство разности;

21) $A \setminus B = A \cap \overline{A} = \varnothing$ – свойство разности;

22) $B \setminus A = B \cap \overline{A} = B \setminus (B \cap A)$– свойство разности;

II. Симметрическая разность и прямое произведение:

1) $A \triangle \varnothing = A$ – свойство нуля;

2) $A \triangle A = \varnothing$ – идемпотентность;

3) $A \triangle B = (A \cup B) \setminus (A \cap B)$ – свойство симметрической разности;

4) $A \triangle B = B \triangle A$ – коммутативность;

5) $(A \triangle B) \triangle C = A \triangle (B \triangle C) = A \triangle B \triangle C$ – ассоциативность;

6) $(A \cup B) \times C = (A \times C) \cup (B \times C)$ – дистрибутивность;

7) $A \times (B \cup C) = (A \times B) \cup (A \times C)$ – дистрибутивность;

8) $(A \cap B) \times C = (A \times C) \cap (B \times C)$ – дистрибутивность;

9) $A \times (B \cap C) = (A \times B) \cap (A \times C)$ – дистрибутивность;

10) $(A \setminus B) \times C = (A \times C) \setminus (B \times C)$ – дистрибутивность;

11) $A \times (B \setminus C) = (A \times B) \setminus (A \times C)$ – дистрибутивность;

Примеры I.2.1.4. Пусть $A=\{1, 2\}$, $B=\{a, b\}$, $C=\{+, -\}$. Тогда дистрибутивность $(A\cup B)\cup C = A\cup(B\cup C) = A\cup B\cup C$ определяется как $(\{1,2\}\cup\{a,b\})\cup\{+,-\} = \{1,2\} \cup (\{a,b\}) \cup \{+,-\}) = \{1,2\} \cup \{a,b\}\cup\{+,-\}$.

Задания I.2.1. Пусть $A=\{1, 2, 4\}$, $B=\{3, 4, 5, 6\}$. Тогда выполните следующие операции:

1) $A\cup\varnothing$;
2) $A\cup A$;
3) $A\cap B$;
4) $A\times B$;
5) $A \setminus B$;
6) $A\cup\overline{A}$;
7) $A\triangle B$;
8) $(A\cup B)\times C$;
9) $(A\setminus B)\times C$.

Вопросы I.2.1:

1. Что такое множество?
2. Как задаются множества?
3. Что такое универсальное множество?
4. Как определяется подмножества?
5. Что такое диаграмма Эйлера-Венна?
6. Как выглядит диаграмма Эйлера-Венна для объединения?
7. Как определяется прямое произведение множеств?
8. Как определяется разность множеств?
9. Как определяется симметричная разность множеств?
10. Как определяется индикаторная функция множества?
11. В чем заключается закон де Моргана?
12. Если $A=\{1,2,3\}$, $B=\{3,4,5\}$, то что означает $\{1,2,3,4,5\}$?
13. Если $A=\{1,2,4\}$, $B=\{4,3,2\}$, то что означает $\{2,4\}$?
14. Если $A=\{1,2,3,4\}$, $B=\{3,4,5,6\}$, то что означает $\{2,4\}$?
15. Что такое дистрибутивность?
16. Что такое ассоциативность?
17. Что такое коммутативность?

I.2.2. Отношения и способы их представления

В этом параграфе будут определены понятие отношения и даны их представления, предложены примеры, даны задания, сформулированы вопросы [14,17,22,27,28,33].

Определения I.2.2.1. Если заданы два множества A и B одного и того же типа, то можно вести следующие отношения:

1) $A = B$: A *равно* B, если A и B состоят из одних и тех же элементов, т.е. A и B являются подмножествами друг друга;

2) $A \subseteq B$: A *содержится в* B, если все элементы A принадлежат B или A равно B, это означает, что A является подмножеством B;

3) $A \subset B$: A *строго содержится в* B, если все элементы A принадлежат B и A не равно B, т.е. некоторые элементы B не принадлежат A, это означает, что A является собственным подмножеством B.

Аналогично можно определить отношения включает $A \supseteq B$ и строго включает $A \supset B$.

Нетрудно заметить, что выше введенные отношения $=$, $\subseteq$ и $\subset$ являются подмножествами прямого произведения $A \times B$.

Таким образом, можно считать, что любое отношение – это некоторое подмножество в прямом произведении, выделяемое определенным законом.

Замечание I.2.2.1. Пустое множество $\emptyset$ является собственным подмножеством любого конечного множества.

Примеры I.2.2.1:

1) если $A=\{a,b,c\}$, $B=\{b,a,c\}$, то $A = B$;
2) если $A=\{1,2,3,4\}$, $B=\{3,1,4,2\}$, то $A \subseteq B$;
3) если $A=\{1,2,3\}$, $B=\{3,1,4,2\}$, то $A \subset B$

Определение I.2.2.2. Пусть $A_1, A_2, \ldots, A_n$ − произвольные множества, не обязательно различные, $n \geq 1$. Тогда *n-арным отношением* на множествах $A_1, A_2, \ldots, A_n$ является подмножество

$$R^n \subseteq A_1 \times A_2 \times \ldots \times A_n,$$

где $\boldsymbol{R}^1$– унарное отношение на $\boldsymbol{A}_1$, $\boldsymbol{R}^2$ – бинарное отношение на $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\boldsymbol{R}^3$ – тренарное отношение на $\boldsymbol{A}_1$, $\boldsymbol{A}_2$, $\boldsymbol{A}_3$ и т.д.

Всякое *унарное отношение* на множестве $\boldsymbol{A}$ является характеристическим свойством некоторого его подмножества. Множество всех унарных отношений на $\boldsymbol{A}$ совпадает с множеством всех подмножеств множества $\boldsymbol{A}$.

Примеры I.2.2.2:

1. Унарное отношение $R_1^1=\{n:\ n\in N \& n<100\}$ определяет множество натуральных чисел, меньших, чем число 100;

2. Унарное отношение $R_2^1=\{n:\ \forall k\in N(n=2*k)\}$ определяет множество четных натуральных чисел;

3. Унарное отношение $R_3^1=\{n:\ \forall k\in Z(n=2*k+1)\}$ определяет множество нечетных целых чисел.

Определения I.2.2.3. *Бинарное отношение* определяется над парами множества и может представлено одним из трех способов:

1) *префиксная запись* – знак отношения вставляется перед участниками бинарного отношения;

2) *инфиксная запись* – знак отношения вставляется между участниками бинарного отношения;

3) *постфиксная запись* – знак отношения вставляется после участниками бинарного отношения.

Бинарные отношения над парами элементов часто представляют с помощью таблиц: строки соответствуют первым элементам пары, столбцы – вторым элементам пары, а наличие отношения между конкретными элементами строки и столбца отмечается специальным знаком, например, знаком «1» или др.

Примеры I.2.2.3:

1. Если $a\in\boldsymbol{A}$ и $b\in\boldsymbol{B}$ находятся в бинарном отношении $\boldsymbol{R}$, то это можно записать как:

$\boldsymbol{R}ab$ – префиксная запись;

$a\boldsymbol{R}b$ – инфиксная запись;

$ab\boldsymbol{R}$ – постфиксная запись.

2. Если множества $\boldsymbol{A} = \{a_1, \ldots, a_r\}$ и $\boldsymbol{B} = \{b_1, \ldots, b_s\}$ находятся в бинарном отношении $\boldsymbol{R}$, то его можно представить с помощью таблицы I.2.2, в которой элементы a_i представляются строками, элементы b_j, – столбцами, а отношения $a_i\boldsymbol{R}b_j$ отмечается "1":

Таблица I.2.2. Двоичное отношение.

$\boldsymbol{R}$	b_1	b_2	…	b_{s-1}	b_s
a_1	1		…	1	
a_2		1	…		
…	…	…	…	…	…
a_{r-1}	1		…	1	
a_r			…		1

Определения I.2.2.4. Говорят, что бинарное отношение $\boldsymbol{R}$ на множестве $\boldsymbol{S}$:

1) *рефлексивно*, если для каждого $s \in \boldsymbol{S}$ имеет место $s\boldsymbol{R}s$;

2) *транзитивно*, если для любых $s,t,u \in \mathbf{S}$ из $s\boldsymbol{R}t$ и $t\boldsymbol{R}u$ следует $s\boldsymbol{R}u$;

3) *симметрично*, если для любых $s, t \in \boldsymbol{S}$ из $s\boldsymbol{R}t$ следует $t\boldsymbol{R}s$;

4) *антисимметрично*, если из $a\boldsymbol{R}b$ и $b\boldsymbol{R}a$ следует $a=b$.

Примеры I.2.2.4:

1. Отношение $>$ над числами не является рефлексивным;
2. Отношения $=, \geq, >$ над числами являются транзитивными;
3. Отношение $=$ над числами является симметричным.

Определение I.2.2.5. Бинарное отношение $\boldsymbol{R}$ называется отношением *эквивалентности*, если оно удовлетворяет свойствам рефлексивности, транзитивности и симметричности.

Каждому отношению эквивалентности на множестве $\boldsymbol{S}$ соответствует единственное разбиение данного множества на смежные классы.

Примеры I.2.2.5. Отношение = в любом числовом множестве будет отношением эквивалентности, т.е. для любых $k, m, n \in \boldsymbol{N}$:

1. Рефлексивность: $n = n$.
2. Транзитивность: из $k < m$ и $m < n$ следует $k < n$.
3. Симметричность: из $k = m$ следует $m = k$.

Определение I.2.2.6. Бинарное отношение $\boldsymbol{R}$ на некотором множестве $\boldsymbol{S}$, удовлетворяющее свойствам рефлексивности, транзитивности и антисимметричности, называется отношением *частичного порядка*.

Примеры I.2.2.6. Отношениями частичного порядка является:

1. Отношение $\subseteq$ для подмножеств некоторого множества;
2. Отношение $\supseteq$ для подмножеств некоторого множества;
3. Отношение = на множестве целых чисел;
4. Отношение $\leq$ на множестве натуральных чисел;
5. Отношение $\geq$ на множестве целых чисел.

Определения I.2.2.7:

1. *Частично упорядоченным множеством* называется множество $\boldsymbol{A}$ с определенным на нем отношением частичного порядка. Точнее говоря, частично упорядоченным множеством называется пара $<\boldsymbol{A}, \boldsymbol{R}>$, где $\boldsymbol{A}$ – множество, а $\boldsymbol{R}$ – отношение частичного порядка на $\boldsymbol{A}$.
2. Элементы $\boldsymbol{a}$ и $\boldsymbol{b}$ частично упорядоченного множества $\boldsymbol{A}$ называются *сравнимыми* относительно частичного порядка $\boldsymbol{R}$ на этом множестве, если имеет место $\boldsymbol{aRb}$ или $\boldsymbol{bRa}$.
3. Частичный порядок на множестве $\boldsymbol{A}$ называется *линейным порядком*, если любые два элемента $\boldsymbol{a}$ и $\boldsymbol{b}$ из $\boldsymbol{A}$ сравнимы относительно частичного порядка $\boldsymbol{R}$.
4. *Линейно-упорядоченным множеством* или *цепью* называется частично-упорядоченное множество, в котором элементы каждой пары сравнимы.

5. Элементов $\boldsymbol{a}$ и $\boldsymbol{b}$ частично упорядоченного множества $\boldsymbol{A}$ называют *несравнимыми*, если между ними не выполнено ни одно отношение частичного порядка. Возможность существования несравнимых элементов объясняет смысл термина *«частично упорядоченное множество»*.

Примеры 1.2.2.7:

1. Множество натуральных чисел $\boldsymbol{N}$ с отношением $\leq$ является частично-упорядоченным, все натуральные числа будут сравнимыми относительно отношения $\leq$, и $\boldsymbol{N}$ есть цепь.

2. Множество всех действительных чисел с отношением = является линейно упорядоченным множеством, если все действительные числа будут сравнимыми относительно отношения =.

3. Пусть $\boldsymbol{A}$ – множество действительнозначных функций на отрезке [0,1] с определенным на нем отношением частичного порядка <, =, >, то элементы $\boldsymbol{f}(x)=x$ и $\boldsymbol{g}(x)=1-x$ будут несравнимы.

Определения I.2.2.8. Пусть $\boldsymbol{A}$ – частично-упорядоченное множество, $\boldsymbol{B}$ – его подмножество, т.е. $\boldsymbol{A} \supseteq \boldsymbol{B}$. Тогда:

1) *нижней гранью* (*верхней гранью*) множества $\boldsymbol{B}$ во множестве $\boldsymbol{A}$ называется элемент $a \in \boldsymbol{A}$, такой, что $a \leq b$ ($b \leq a$) для любого $b \in \boldsymbol{B}$;

2) элемент $a \in \boldsymbol{A}$ называется *наименьшим* (*наибольшим*) во множестве $\boldsymbol{A}$, если a есть нижняя (верхняя) грань самого $\boldsymbol{A}$;

3) элемент $a \in \boldsymbol{A}$ называется *минимальным* (*максимальным*) во множестве $\boldsymbol{A}$, если не существует $b \in \boldsymbol{A}$, такого, что $b < a$ ($a < b$).

Наименьший (наибольший) элемент множества $\boldsymbol{A}$ является его единственным минимальным (максимальным) элементом.

Примеры I.2.2.8:

1. Множество всех подмножеств множества $\boldsymbol{A}$ имеет наименьший элемент $\emptyset$ и наибольший элемент само $\boldsymbol{A}$.

2. Множество $\boldsymbol{N}$ натуральных чисел имеет наименьший элемент 1 и не имеет наибольшего элемента.

3. Множество $\boldsymbol{Z}$ всех целых чисел не имеет наименьшего, наибольшего, минимального и максимального элемента.

Элементы множеств $\boldsymbol{A}$ и $\boldsymbol{B}$ находятся во *взаимно-однозначном соответствии*, если каждому элементу $a\in\boldsymbol{A}$ по некоторому закону сопоставлен один и тот же элемент $b\in\boldsymbol{B}$, причем каждый $b\in\boldsymbol{B}$ оказывается сопоставленным одному и тому же $a\in\boldsymbol{A}$.

Множества $\boldsymbol{A}$ и $\boldsymbol{B}$ являются *эквивалентными* (*равномощными*), если можно установить взаимно-однозначное соответствие между их элементами.

Замечание I.2.2.2. Бинарное отношение может задаться тройкой множеств $<\boldsymbol{R}, \boldsymbol{A}, \mathbf{B}>$, где $\boldsymbol{R}\subseteq\boldsymbol{A}\times\boldsymbol{B}$ – график отношения и записываться $(a, \underline{b})\in\boldsymbol{R}$ или $a\boldsymbol{R}b$. Тогда можно определить:

Область определения: $\mathbf{Dom}\ \boldsymbol{R} = \{x\in\boldsymbol{A}:\ \exists y\in\boldsymbol{B}(x, y)\in\boldsymbol{R}\}$;

Область значения: $\mathbf{Run}\ \boldsymbol{R} = \{y\in\boldsymbol{B}:\ \exists x\in\boldsymbol{A}(x, y)\in\boldsymbol{R}\}$;

Обратное отношение: $\boldsymbol{R}^{-1} = \{(y, x)\in\boldsymbol{B}\times\boldsymbol{A}:\ (x, \underline{y})\in\boldsymbol{R}\}$;

Композиция отношения: $\boldsymbol{R}\subseteq\boldsymbol{A}\times\boldsymbol{B},\ \boldsymbol{S}\subseteq\boldsymbol{B}\times\boldsymbol{C},$

$$\boldsymbol{R}\cdot\boldsymbol{S}=\{(x, z)\in\boldsymbol{A}\times\boldsymbol{C}:\ \exists y\in\boldsymbol{B}\ [(x\boldsymbol{R}y)\ \&(y\boldsymbol{S}z)]\}.$$

Определение I.2.2.9. Бинарное отношение $\boldsymbol{f}\subseteq\boldsymbol{X}\times\boldsymbol{Y}$ называется *функцией* из $\boldsymbol{X}$ в $\boldsymbol{Y}$, если $\mathbf{Dom}\ \boldsymbol{R} = \boldsymbol{X}$ и $(x, y)\in\boldsymbol{f},\ (x, z)\in\boldsymbol{f}\ \Rightarrow y=z$.

Функция $\boldsymbol{f}$: $\boldsymbol{X}\to\boldsymbol{Y}$ называется:

1) *сюръективной*, если для любого $y\in\boldsymbol{Y}$ существует $x\in\boldsymbol{X}$ такой, что $y=\boldsymbol{f}(x)$, т.е. $\forall y\in\boldsymbol{Y}\ \exists x\in\boldsymbol{X}(y=\boldsymbol{f}(x))$;

2) *инъективной*, для любых $x_1,\ x_2\in\boldsymbol{X}$ из того, что $x_1\neq x_2$ следует $\boldsymbol{f}(x_1)\neq\boldsymbol{f}(x_2)$, т.е. $\forall x_1\in\boldsymbol{X}\ \forall x_2\in\boldsymbol{X}\ (x_1\neq x_2\Rightarrow\boldsymbol{f}(x_1)\neq\boldsymbol{f}(x_2))$;

3) *биективной*, если она сюръективна и инъективна.

Любую бинарную функцию можно связать с тренарным отношением, например, если задана бинарная функция $\boldsymbol{f}(x,y)$, то ее можно связать с тренарным отношением $\boldsymbol{R}^3(x,y,z)$ так, чтобы $z=\boldsymbol{f}(x,y)$.

Примеры I.2.2.9. Пусть $x,\ y,\ z\in\boldsymbol{N}$ – натуральные числа и задана бинарная функция $\boldsymbol{f}(x,y)=z$, тогда:

1) если x есть 3, y есть 5, z есть 8, а $\boldsymbol{f}$ есть операция сложение +, то вместо записи $\boldsymbol{f}(x,y)=z$ можно писать 3+5=8 и с ней связать тренарное отношение ***Add***(3, 5, 8), для которого справедливо 8 = 3+5.

2) если x есть 8, y есть 2, z есть 4, а $\boldsymbol{f}$ есть операция деление «:», то вместо записи $\boldsymbol{f}(x,y)=z$ можно писать 8:2 = 4 и с ней связать тренарное отношение ***Dev***(8, 2, 4), для которого справедливо 4 = 8:2.

Ясно, что для некоторых $x, y \in \boldsymbol{N}$ результатом выполнения операции деления является не целое число и для них тренарное отношение ***Dev***(x, y, z) не выполняется. Поэтому для таких случаев необходимо определить дополнительные условия о результатах.

Определение I.2.2.10. *Транзитивным замыканием* отношения $\boldsymbol{R}$ на множестве $\boldsymbol{A}$ называется пересечение всех транзитивных отношений, содержащих $\boldsymbol{R}$ как подмножество (иначе, минимальное транзитивное отношение, содержащее $\boldsymbol{R}$ как подмножество).

Транзитивное замыкание существует для любого отношения. Для этого отметим, что пересечение любого множества транзитивных отношений транзитивно. Более того, обязательно существует транзитивное отношение, содержащее $\boldsymbol{R}$ как подмножество.

Транзитивное замыкание обладает следующими свойствами:

1) Транзитивное замыкание рефлексивного отношения рефлексивно, т.к. транзитивное отношение содержит исходное отношение;

2) Транзитивное замыкание симметричного отношения симметрично. Действительно, пусть имеется транзитивное отношение $a\boldsymbol{R}b$, значит существуют $x_1, x_2, \ldots, x_n$ такие, что $a\boldsymbol{R}x_1, x_1\boldsymbol{R}x_2, \ldots, x_n\boldsymbol{R}b$. Но из симметричности отношения $\boldsymbol{R}$ следует $b\boldsymbol{R}x_n, x_n\boldsymbol{R}x_{n-1}, \ldots, x_1\boldsymbol{R}a$, следовательно, $b\boldsymbol{R}a$.

3) Транзитивное замыкание не сохраняет антисимметричность, например, для отношения $\{(a,b), (b,c), (c,a)\}$ на множестве $\{a, b, c\}$.

4) Транзитивное замыкание транзитивного отношения - оно само.

Отношение $\boldsymbol{R}^* = \boldsymbol{R}^+ \cup \boldsymbol{R}^0$, где $\boldsymbol{R}^0 = \{(\varepsilon, \varepsilon): \varepsilon \in \boldsymbol{A}\}$ иногда называют рефлексивно-транзитивным замыканием, хотя часто под "транзитивным

замыканием" подразумевается именно $\boldsymbol{R^*}$. Обычно различия между этими отношениями не являются значительными.

Примеры I.2.2.10:

1. Для любых $a, b, c \in \boldsymbol{N}$ из справедливости отношений $a<b$ и $b<c$ следует отношение $a<c$.
2. Для любых $x, y \in \boldsymbol{N}$ из справедливости отношения $x=y$ следует отношение $y=x$.
3. Если $\boldsymbol{A}$ – множество городов, и на них задано отношение $x\boldsymbol{R}y$, означающее "существует автобусный маршрут из x в y", то транзитивным замыканием этого отношения будет отношение "существует возможность добраться автобусом из x в y".

Задания I.2.2. При $\boldsymbol{A}=\{a,b,c,d\}$, $\boldsymbol{B}=\{b,d\}$, $\boldsymbol{C}=\{c\}$ определить:

1. $\boldsymbol{A}\triangle\boldsymbol{B}$;
2. $\boldsymbol{A}\cup\boldsymbol{B})\times\boldsymbol{C}$;
3. $\boldsymbol{A}\times(\boldsymbol{B}\cup\boldsymbol{C})$;
4. $(\boldsymbol{A}\cap\boldsymbol{B})\times\boldsymbol{C}$;
5. $\boldsymbol{A}\times(\boldsymbol{B}\cap\boldsymbol{C})$;
6. $\boldsymbol{A}\cap(\boldsymbol{B}\cup\boldsymbol{C})$;
7. $\boldsymbol{A}\cup(\boldsymbol{B}\cap\boldsymbol{C})$.
8. $\boldsymbol{A}\setminus(\boldsymbol{B}\cup\boldsymbol{C})$;
9. $(\boldsymbol{A}\setminus\boldsymbol{B})\cap\boldsymbol{C}$.
10. $(\boldsymbol{A}\triangle\boldsymbol{B})\cap\boldsymbol{C}$;

Вопросы I.2.2:

1. Сколько множеств участвуют в бинарном отношении?
2. В чем заключается частичная упорядоченность множества?
3. Как определяется функция?
4. Как определяется композиция отношений?
5. Из чего состоит мощность объединения двух множеств?
6. Как определяется эквивалентность двух множеств?
7. Какие множества являются равномощными?
8. Какие элементы являются несравнимыми?

9. Что такое эквивалентность двух множеств?
10. Что такое транзитивное замыкание отношения?

I.2.3. Символы, цепочки и операции над ними

В этом параграфе будут рассмотрены понятия символа и цепочки, определены операции над ними и разбираются свойства этих операций, а также будут предложены примеры, даны задания, сформулированы вопросы [1-9,11-18,21,25,27-32].

Определения I.2.3.1:

1. Символом называется минимальная синтаксическая единица, которая используется при определении любого языка.
2. Символами могут быть терминалы - само определенные постоянные величины или нетерминалы - определенные через дугие символы переменные величины.
3. Терминалами являются буквы, цифры и специальные знаки (операции, скобки, препинания и т.п.).
4. Нетерминалами являются морфемы, лексемы, слова, предложения и т.п., обозначающие некоторые понятия.

Соглашения I.2.3.1:

1. Имя терминала задается представлениям своего значения.
2. Имя нетерминала задается строчными греческими буквами с индексами или без них, которыми могут служить цифры или строчные латинские буквы.

Для работы с заданными символами нужно применить *операции*, определенные над ними, а для применения любой операции нужно знать её *определение*, *обозначение* и *свойства*.

С помощью символьных операций строятся символьные выражения. Для обработки символьных выражений используются свойства операций, упрощающие сложные символьные выражения, сокращая количество операции в них и облегчая их выполнения.

Над символами определены операции *строения*, позволяющие из двух символьных величин получить одну символьную величину.

Самой простой операцией строения является *конкатенация* (*сцепление*). Обычно эта операция обозначается знаком ‘·’ и ее можно определить над двумя символьными величинами $\boldsymbol{\alpha}$ и $\boldsymbol{\beta}$. Результатом

операции сцепления к символу α символа β будет символ γ, значение которого является цепочка значений α и β, и это записывается как $\alpha \cdot \beta = \gamma$. Следует отметить, что значениями символов α и β могут быть как, один терминал, так и цепочки терминалов.

Из символов с помощью операции конкатенации строятся различные цепочки (*слова*, *строки*). Операция конкатенация также может быть определена над цепочками, результатом которой опять будет цепочка.

Для полноты во множество символьных величин вводится абстрактная величина, называемая «пустая цепочка». Пустая цепочка не содержит ни одного символа и обозначается как ‘ε’.

Замечания I.2.3.2:

1. В символьных выражениях знак операции сцепления ‘·’ не пишется, он пишется только при определении свойств операции, чтобы не спутать ее с другими операциями.
2. Пустая цепочка цепочка во множестве символьных величин выполняет такую же роль, как число ноль во множестве числовых величин.

Следующей операцией строения является *дизъюнкция* (*выбор*), обозначаемая знаком ‘∨’. Результатом операции выбора над двумя символами α и β будет символ γ, значение которой образуется из занчения α или значения β, и это записывается как $\alpha \vee \beta = \gamma$. Например, если значением α является ‘робот’, а значением β – ‘компьютер’, то значением γ будет ‘робот’ или ‘компьютер’.

Над непустой цепочкой α можно определить операцию *удаление* $\alpha^0=\varepsilon$, которая не определена для пустой цепочки.

К операции строения относится также и операция *итерация*, которая обозначается знаком ‘*’.

Для любой символьной величины α операция итерация ‘*’ определяется как

$$\alpha^*=\alpha^0 \vee \alpha^1 \vee \alpha^2 \vee \alpha^3 \vee \ldots \vee \alpha^k \vee \ldots,$$

где $\alpha^0=\varepsilon$, $\alpha^1=\alpha$, $\alpha^k=\underbrace{\alpha\cdot\alpha\cdot\ldots\cdot\alpha}_{k}$, $k=2,3,\ldots$

Чтобы правильно построить символьное выражение с помощью определенных выше операций нужно знать порядок выполнения: * – первая, ^{0} и $\cdot$ – вторая, $\vee$ – третья. Для установления иного порядка выполнения применяются скобки.

Рассмотренные выше операции можно назвать *базовыми символьными операциями*. С помощью их можно определить любые сложные символьные операции, которые потребуются при обработке символьных выражений. Но для этого нужно знать алгебраические свойства базовых символьных операций.

Пусть заданы произвольные непустые цеопочки α, β, γ и пустая цепочка ε. Тогда алгебраические свойства базовых символьных операций выражаются следующими равенствами:

$\alpha\cdot\varepsilon=\varepsilon\cdot\alpha=\alpha$;

$\alpha\cdot\beta\neq\beta\cdot\alpha$ - для некоторых α, β ;

$\alpha\cdot(\beta\cdot\gamma)=(\alpha\cdot\beta)\cdot\gamma=\alpha\cdot\beta\cdot\gamma$;

$\alpha^0=\varepsilon$;

$\alpha\alpha^0=\alpha^0\alpha=\alpha$;

$\alpha\vee\beta=\beta\vee\alpha$;

$\alpha\vee(\beta\vee\gamma)=(\alpha\vee\beta)\vee\gamma=\alpha\vee\beta\vee\gamma$;

$\alpha\cdot(\beta\vee\gamma)=\alpha\cdot\beta\vee\alpha\cdot\gamma$;

$(\alpha\vee\beta)\cdot\gamma=\alpha\cdot\gamma\vee\beta\cdot\gamma$;

$\alpha\vee\alpha=\alpha$;

$\varepsilon^*=\varepsilon$;

$(\alpha^*)^*=\alpha^*$;

$\alpha^*=\alpha\vee\alpha^*$.

Используя эти свойства базовых символьных операций можно выразить (определить) сложные символьные операции.

Определения I.2.3.2:

1. Число символов заданной цепочки ω называется *длиной цепочки* ω, причем каждый символ считается столько раз, сколько раз он встречается в ω, она обозначается как |ω|. Длина пустой цепочки равна нулю, т.е. |ε| = 0.

2. Символ *a входит* в цепочку ω, если существуют такие цепочки η и ξ, что ω = $\eta a \xi$. Количество вхождений символа *a* в цепочку ω обозначается через $|\omega|_a$.

3. Цепочка τ входит в цепочку ω, если существуют такие цепочки η и ξ, что ω = $\eta\tau\xi$ и обозначается τ ⊆ ω, т.е. $\tau \subseteq \omega \rightleftharpoons \exists\eta\exists\xi(\omega = \eta\tau\xi)$.
Количество вхождений цепочки τ в цепочку ω обозначается как $|\omega|_\tau$.

4. *Обратной цепочкой* заданой цепочки ω называется цепочка, которая составлена из символов цепочки ω в обратном порядке. Обратная цепочка получается применением операции *обращения* к исходной цепочке ω и обозначается как ω^R.

Примеры I.2.3:

1. Если значением цепочки $\boldsymbol{\alpha}$ является ‘қазах’, а значением цепочки $\boldsymbol{\beta}$ –‘стан’, то значением цепочки $\boldsymbol{\gamma}$ ‘қазахстан’.
2. Пусть $\boldsymbol{\alpha}$ = ‘моя родина’ и $\boldsymbol{\beta}$ = ‘ қазахстан’, тогда значением $\boldsymbol{\gamma}$ будет ‘моя родина қазахстан’.
3. Для цепочки *abbac* запишем |*abbac*| = 4, $|abbac|_a = 2$, $|abbac|_b = 2$, $|abbac|_c = 1$.
4. Справедливы равенства: $ab^3 = abbb$, $a^2b^2 = aabb$, $(ab)^3 = ababab$.
5. Если задана цепочка *aaa*, то $aaa^\circ = aa^\circ a = a^\circ aa = aa$.
6. Если ω = *abcd*, то $\omega^R = dcba$.
7. Если $\alpha = aa$, $\beta = abaacd$, $\xi = cd$, $\zeta = ab$, то $\beta = \zeta\alpha\,\xi$, т.е. $\alpha \subseteq \beta$.
8. Если ω = *abcada*, то $|\omega|_a = 3$.
9. Если ω = *abcabcdabda* и σ =*ab*, то $|\omega|\sigma = 2$.
10. Пусть задана цепочка $\boldsymbol{a \cdot b \cdot c}$. Требуется переставить символы этой цепочки в обратном порядке, т.е. получить обращение $(\boldsymbol{a \cdot b \cdot c})^R = \boldsymbol{c \cdot b \cdot a}$. Для этого используя свойства базовых операций удаления и сцепления можно выразить сложную символьную операцию *обращения* с помощью:

$$(a\cdot b\cdot c)^{R}=a^{0}\cdot b\cdot c=\varepsilon\cdot b^{0}\cdot c=\varepsilon\cdot\varepsilon\cdot c=c\cdot b=c\cdot b\cdot a \quad .$$

Задания I.2.3:

1. При $\alpha = abcd$, $\beta = ecbaf$, $\gamma = def$ определить:

1) все подцепочки цепочки β;
2) все подцепочки цепочки $\alpha\beta$;
3) все подцепочки цепочки $\beta\gamma$;
4) все подцепочки цепочки α ;
5) все подцепочки цепочки $\alpha\gamma$;
6) $\alpha\cdot\beta$;
7) $\alpha\vee\beta$;
8) $\alpha\cdot(\beta\vee\gamma)$;
9) $(\alpha\vee\beta)\cdot\gamma$

2. Постройте символьное выражение из цепочек α и β, чтобы его значение было равно цепочки γ для их следующих значений:

$\alpha=abcd$, $\beta=ecbaf$, $\gamma=def$;
$\alpha=12345$, $\beta=6789$, $\gamma=28f$;
$\alpha=abcd$, $\beta=123456$, $\gamma=c2d5$.

3.Найти $|\omega|_\sigma$, если $\alpha=cmnkabmncdmn$, $\sigma = mn$.

Вопросы I.2.3:

1. Что такое символ?
2. В чем разница между терминалом и нетерминалом?
3. Как задаются имена нетерминалов?
4. Чему равна длина цепочки?
5. Сто такое пустая цепочка?
6. Как определяется операция конкатенация над символами?
7. Как опрелеляется операция дизъюнкция над символами?
8. Как опрелеляется операция итерация над символами?
9. Как определяется вхождение символа в цепочку?
10. Как определяется вхождение цепочки в другую цепочку?
11. Что такое длина цепочки?
12. Как строится обратная цепочка?
13. Для каких цепочек выполняется закон коммутативности для операции сцепления?

I.2.4. Алфавит, слово, язык и операции над языками

В этом параграфе будут рассмотрены понятия алфавита, символа, слово и языка, определены операции над ними и разбираются их свойства, предложены примеры, даны задания, сформулированы вопросы [1-9,11-18,21,25,27-32].

Определения I.2.4.1:

1. *Алфавитом* является конечное непустое множество элементов, называемых символами (буквами). В дальнейшем, алфавит будет обозначаться прописными (заглавными) латинскими буквами, а элементы - строчными (маленькими).

2. *Словом* (*цепочкой*) в заданном алфавите называется конечная последовательность элементов этого алфавита. В дальнейшем, слова будут обозначаться строчными греческими буквами.

3. Слово, не содержащее ни одного символа называется *пустым словом*, и обозначается греческой буквой ε.

4. *Длина слова* ω задается числом его символов и обозначается $|\omega|$. Причем каждый символ считается столько раз, сколько он встречается в ω. Длина пустого слова равна нулю, т.е. $|\varepsilon|=0$.

5. Если α и β есть слова в алфавите ***T***, то слово $\alpha\cdot\beta$ называется *конкатенацией* (*сцеплением*) слов α и β, которая получается в результате приписывания слова β в конец слова α. Обычно в записи знак · опускают и пишут просто $\alpha\beta$.

6. Если α – слово и $n\geq 0$ – целые положительное число, то через $\alpha^0 \rightleftharpoons \varepsilon$ и $\alpha^n \rightleftharpoons \underbrace{\alpha\cdot\alpha\cdot\ldots\cdot\alpha}_{n}$ (знак $\rightleftharpoons$ читается “равно по определению”).

7. Если τ и ω есть слова в алфавите ***T*** и для некоторого слова ξ в ***T*** выполняется ω = τξ, то слово τ называется *префиксом* (*началом*) слова ω и обозначается как $\alpha\sqsubset\beta \rightleftharpoons \exists\xi(\beta = \alpha\xi)$.

8. Если τ и ω есть слова в алфавите ***T*** и для некоторого слова ζ в ***T*** имеет место ω = ζτ, то слово τ называется *постфиксом* (*концом*) слова ω и обозначается как: $\alpha\sqsupset\beta \rightleftharpoons \exists\zeta(\beta = \zeta\alpha)$.

9. Если τ и ω есть слова в алфавите ***T*** и для некоторых слов ζ и ξ выполняется ω = ζτξ, то слово τ называется *подсловом* (*подцепочкой*) слова ω и обозначается как: $\tau \subseteq \omega \rightleftharpoons \exists\zeta\, \exists\xi(\omega = \zeta\tau\xi)$.

10. Если τ и ω есть слова в алфавите ***T*** и τ является подсловом слова ω, то через $|\omega|_\tau$ обозначается количество вхождений слово τ в слове ω.

Замечание I.2.4.1.

Для любого непустого слово ω в алфавите ***T*** и пустого слово ε выполняются отношения $\varepsilon \Subset \omega$ и $\varepsilon \Supset \omega$.

Примеры I.2.4.1: Пусть задан алфавит ***T***={*a,b,c,d*}. Тогда:

1. *a, bb, ccc, abcd, dcba* и другие последовательности, состоящие из букв *a,b,c,d* являются словами в ***T***.
2. $|a|=1$, $|bb|=2$, $|ccc|=3$, $|abcd|=4$.
3. Если $\alpha = ccc$ и $\beta = dddd$, то $\alpha\beta = cccdddd$.
4. $ab^2 = abb$, $(ab)^3 = ababab$.
5. Если $\alpha = ab$, $\beta = abcd$, $\xi = cd$, *то* $\beta = \alpha\xi$, т.е. $\alpha \Subset \beta$.
6. Если $\alpha = cd$, $\beta = abcd$, $\zeta = ab$, *то* $\beta = \zeta\alpha$, т.е. $\alpha \Supset \beta$.
7. $\varepsilon \Subset abcd$, $a \Subset abcd$, $ab \Subset abcd$, $abc \Subset abcd$, $abcd \Subset abcd$.
8. $abcd \Supset \varepsilon$, $abcd \Supset d$, $abcd \Supset cd$, $abcd \Supset bcd$, $abcd \Supset abcd$.
9. Если $\tau = ca$ и $\xi = da$, то $\tau \Subset \omega$ и $\omega = cada$.
10. Если $\tau = 12$ и $\zeta = 11$, то $\tau \Supset \omega$ и $\omega = 1112$.
11. Если $\tau = bbb$, $\zeta = aaa$ и $\xi = ccc$, то $\tau \subseteq \omega$ и $\omega = aaabbbccc$.
12. Если $\omega = abcdabcd$ и $\tau = cd$, то $\tau \subseteq \omega$ и $|\omega|_\tau = 2$.

Если ***T*** – алфавит, то множество всех цепочек (слов) конечной длины в этом алфавите определяется как

$$\boldsymbol{T^*} \rightleftharpoons \bigcup_{k=0}^{\infty} \boldsymbol{T}^k,$$

где $\boldsymbol{T^0} \rightleftharpoons \{\varepsilon\}$ – множество цепочек длины 0, $\boldsymbol{T^k} \rightleftharpoons \underbrace{\boldsymbol{T}\cdot\boldsymbol{T}\cdot\ldots\cdot\boldsymbol{T}}_{\boldsymbol{k}}$ – множество цепочек длины *k*, $k \geq 1$ – натуральные числа.

Через T^+ обозначается множество всех возможных непустых цепочек, т.е. $T^+ = T^*\backslash\{\varepsilon\}$. Например, если $T = \{a\}$, то множество T^+ определяется как $T^+ = \{a, aa,, aaa, \ldots\}$.

Очевидно, что не все цепочки из множества T^+ могут быть осмысленными единицами (словами, словосочетаниями, фразами, предложения и текстами) некоторого языка. Заметим, что осмысленными единицами языка могут быть только те цепочки, которые удовлетворяют грамматическим правилам и имеют семантические значения. Поэтому любой конкретный язык L является собственным подмножеством множества T^+, т.е. $L \subset T^+$. Например, если в качестве элементов алфавита T взять казахских букв и различные специальные знаки, то казахский язык будет собственным подмножеством множества T^+, в котором содержатся только слова, словосочетания, фразы, предложения и тексты, имеющие смысл в казахском языке.

Поскольку каждый язык является множеством цепочек конечной длины в заданном алфавите, то можно рассматривать заданных над одним и тем же алфавитом операции *объединения*, *пересечения*, *разности*, *дополнения*, *прямого произведения*, *симметрической разности*, *конкатенации* и *итерации* языков.

Определения I.2.4.2. Пусть заданы два языка L_1 и L_2 в алфавите T, т.е. $L_1 \subseteq T^+$ и $L_2 \subseteq T^+$, а также U – универсум, тогда можно определить:

1. Объединение: $L_1 \cup L_2 \rightleftharpoons \{x: x \in L_1 \vee x \in L_2\}$;
2. Пересечение: $L_1 \cap L_2 \rightleftharpoons \{x: x \in L_1 \,\&\, x \in L_2\}$;
3. Разность: $L_1 \setminus L_2 \rightleftharpoons \{x: x \in L_1 \,\&\, x \notin L_2\}$;
4. Дополнение: $\overline{L} \rightleftharpoons \{x: x \in U \,\&\, x \notin L\}$;
5. Прямое произведение: $L_1 \times L_2 \rightleftharpoons \{(a, b): a \in L_1 \,\&\, b \in L_2\}$;
6. Симметрическая разность: $L_1 \triangle L_2 \rightleftharpoons \{x: x \in (L_1 \setminus L_2) \vee x \in (L_2 \setminus L_1)\}$;
7. Конкатенация: $L_1 \cdot L_2 \rightleftharpoons \{a \cdot b: a \in L_1 \,\&\, b \in L_2\}$;
8. Итерация – звездочка Клини (Kleene star): $L^* \rightleftharpoons \bigcup_{k=0}^{\infty} L^k$, где

$L^0 \rightleftharpoons \{\varepsilon\}$, $L^k \rightleftharpoons \underbrace{L \cdot L \cdot \ldots \cdot L}_{k}$, $k \geq 1$.

Примеры I.2.4.2: Пусть заданы языки $L_1 = \{aa,bb\}$, $L_2 = \{cc,dd\}$

1. Если $L_1 = \{aa, bb\}$, $L_2 = \{cc, dd\}$, то
$$L_1 \cdot L_2 = \{aacc, aadd, bbcc, bbdd\}.$$
2. Если $L = \{a^k b a^l: \ 0 < k < l\}$, то
$$L^2 = \{a^k b a^l b a^m: \ \ 0 < k < l - 1, m > 1\}.$$
3. Если $T = \{a, b\}$, $L = \{aa, ab, ba, bb\}$, то
$$L^* = \{\tau \in T^*: \ \ |\tau| = 2\}, \ L^* = \{\tau \in T^*: \ \ |\tau|_a = 1 \ \& \ |\tau|_b = 1\}.$$

Замечания I.2.4.2:

1. Множество слов конечной длины в алфавите T является частично упорядоченным множеством с отношением $\leq$ ($\geq$) и все его цепочки сравнимы по длине относительно $\leq$ ($\geq$).
2. Множество слов конечной длины в алфавите T является частично упорядоченным множеством с отношением $\subseteq$ ($\supseteq$) и все подмножества слов в алфавите T сравнимы относительно $\subseteq$ ($\supseteq$).

Определения I.2.4.3. Пусть $L \subseteq T^*$, тогда можно ввести следующие обозначения и понятия:

1. L^R – операция *обращение* над языком L определяется как $L^R = \{\tau^R: \ \tau \in L\}$.
2. Pref(L) – *множество префиксов* языка L определяется как $\text{Pref}(L) \rightleftharpoons \{\tau: \ \exists\xi(\xi \in L \ \& \ \tau \sqsubset \xi)\}$, где $\sqsubset$ - отношение префикса.
3. Suf(L) – *множество постфиксов* языка L определяется как $\text{Suf}(L) \rightleftharpoons \{\tau: \ \exists\zeta(\zeta \in L \ \& \ \tau \sqsupset \zeta)\}$, где $\sqsupset$ - отношение постфикса.
4. Substr(L) – *множество всех подслов* (*подцепочек*) языка L определяется как $\text{Substr}(L) \rightleftharpoons \{\tau: \ \forall\xi(\xi \in L \ \& \ \xi \neq \varepsilon \Rightarrow \tau \subseteq \xi)\}$.
5. Функция f: $K \rightarrow L$ называется *биекцией*, если каждый элемент множества L является образом ровно одного элемента множества K относительно функции f.
3. Множества K и L называются *равномощными*, если существует биекция из K в L.

Примеры I.2.4.3. Пусть задан язык:

1. $\boldsymbol{L}$= {$a^m ba^n$: $m \le n$} является языком в алфавите {a, b}, тогда множество всех его подцепочек содержит цепочки b, ba, aba, baa, $abaa$, $baaa$, $aabaa$, $abaaa$ и т. д.

2. $\boldsymbol{L}$= {ab^n: $n \ge 0$} является языком в алфавите {a, b}, тогда множество всех его подцепочек содержит цепочки a, ab, abb, $abbb$, $abbbb$, $abbbbb$ и т. д.

3. $\boldsymbol{L}$= {abc, a} является языком в алфавите {a, b, c}, тогда множество всех подслов этого языка $\boldsymbol{L}$ будет равно

$$\text{Substr}(\boldsymbol{L}) = \{\varepsilon, a, b, c, ab, ac, bc, abc\}.$$

Задания I.2.4:

1. Пусть заданы $\boldsymbol{T} = \{a, b\}$ и $\boldsymbol{L} = \{aa, ab\}$. Найти $\boldsymbol{L}^3$.
2. Пусть задан алфавит $\boldsymbol{T} = \{a, b, c, d\}$ и язык

$$\boldsymbol{L} = \{\tau \in \boldsymbol{T}^*: \ |\tau|_a = 1 \ \& \ |\tau|_b = 1\}.$$

Верно ли, что $abcdcacdcabbacba \in \boldsymbol{L}^*$.

3. Существует ли такой язык $\boldsymbol{L}$, что выполняется неравенство

$$\boldsymbol{L}^* \neq \{x^n: \ x \in \boldsymbol{L}, n \ge 0\}.$$

4. Существует ли такой язык $\boldsymbol{L}$, что выполняется неравенство

$$(\boldsymbol{L}^{\boldsymbol{R}})^* \neq (\boldsymbol{L}^*)^{\boldsymbol{R}} ?$$

5. Перечислить слова языка $\boldsymbol{L_1} \cap \boldsymbol{L_2}$, где $\boldsymbol{L_1} = \{(ab)^n: \ n \ge 0\}$ и $\boldsymbol{L_2} = \{a^m b^m: \ m \ge 0\}$.

6. Пусть $\boldsymbol{T} = \{a, b, c\}$. Показать равенство следующих языков

$$\boldsymbol{L_1} = \{(abc)^n a: \ n \ge 2\} \text{ и } \boldsymbol{L_2} = \{ab(cab)^n ca: \ n \ge 1\}.$$

7. Пусть $\boldsymbol{T} = \{a, b, c\}$. $\boldsymbol{L_1} = \{\tau \in \boldsymbol{T}: \ |\tau| = 4\}$ и $\boldsymbol{L_2} = \{\tau \in \boldsymbol{T}^*: \ |\tau|_c = 1\}$. Вычислить число цепочек языка $\boldsymbol{L}_1 \setminus \boldsymbol{L}_2$.

8. Пусть заданы языки $\boldsymbol{L_1}$ и $\boldsymbol{L_2}$. Выяснить равномощность языков $\boldsymbol{L}_1 \cdot (\boldsymbol{L}_2^{\boldsymbol{R}})$ и $\boldsymbol{L}_1^{\boldsymbol{R}} \cdot \boldsymbol{L}_2$.

9. Для языков $\boldsymbol{L}_1$ и $\boldsymbol{L}_2$ определите результат их конкатенации и объединения:

$\boldsymbol{L_1}$= {d, de, dee} и $\boldsymbol{L_2}$= {ε, d, e, de, d};

$\boldsymbol{L_1}$= {ε, d, de, dee} и $\boldsymbol{L_2}$= { d, e, dee, d};

$\boldsymbol{L_1}$= {ε, d, e, de, ded} и $\boldsymbol{L_2}$= {ε, d, e, de, ed};

$\boldsymbol{L_1}$= {d, e,dd, de, ee, ded} и $\boldsymbol{L_2}$= {d, e, dd, de, ed}.

$\boldsymbol{L}_1$= {*d, e, ded, ddde, eedd*} и $\boldsymbol{L}_2$= {*d, e, ddd, ded, eee*}.

$\boldsymbol{L}_1$= { ε, *d, e, ddd, eee*} и $\boldsymbol{L}_2$= {*ddee, eedd*}.

10. Пусть задан язык $\boldsymbol{L}$ = {*dcc, dcd, ddc, ddd*}. Нужно определить какой из следующих языков получается из итерации $\boldsymbol{L}^*$?

{ω: $\omega = d\xi$ и $|\omega|$ делится на 3} $\cup${ε};

{ω: $\omega = d\xi$ и $|\omega| \geq$ } $\cup${ε};

{ω: $\omega = \omega_1 \omega_2 \omega_3 \ldots \omega_{3n}$ и $\omega_{3i+1} = d$ для всех $i < n$ } $\cup${ε};

{ω: $\omega = d\xi$ и $|\omega| \geq 12$}.

{ω: $\omega = \xi d$ и $|\omega| \geq 4$}.

{ω: $\omega = \xi d$ и $|\omega|$ делится на *4*}.

11. Пусть $\boldsymbol{L}$ = {*abcd, ad*} является языком в алфавите {*a, b, c, d*}. Найти множество всех подслов этого языка Substr($\boldsymbol{L}$).

12. Пусть $\boldsymbol{L}$ = {$a^k b^m c^n$: $k \leq m \leq n$} является языком в алфавите {*a,b,c*}. Найти множество всех подслов этого языка Substr($\boldsymbol{L}$).

Вопросы I.2.4:

1. Что такой префикс цепочки?
2. Что такой постфикс цепочки?
3. Если ω = *abcdabefabhgabik* и τ = *ab*, то чему равна $|\omega|_\tau$?
4. Как определяется множество всех цепочек?
5. Что такой язык в заданном алфавите?
6. Как определяется $\overline{L}$?
7. Как определяется $\boldsymbol{L_1} \setminus \boldsymbol{L_2}$?
8. Как определяется $\boldsymbol{L_1} \triangle \boldsymbol{L_2}$?
9. Как определяется $\boldsymbol{L_1} \cdot \boldsymbol{L_2}$?
10. Как определяется $\boldsymbol{L}^+$
11. Чему равно $\boldsymbol{L}^0$?
12. Чему равно $\boldsymbol{L}^k$?
13. Как строится обращение языка $\boldsymbol{L}^R$?
14. Существует ли такой язык $\boldsymbol{L}$, что выполняется $(\boldsymbol{L}^R)^* \neq (\boldsymbol{L}^*)^R$?
15. Как получается префикс языка Pref($\boldsymbol{L}$)?
16. Как получается постфикс языка Suf($\boldsymbol{L}$)?

17. Как определяется Substr(***L***)?

I.3. Механизмы определения языков

I.3.1. Порождающие механизмы языков

В этом параграфе будут рассмотрены формальные грамматики – механизмы порождения языков, отношения выводимости и языка, порождаемого формальной грамматикой, предложены примеры, даны задания, сформулированы вопросы [1-9,11-18,21,25,27-32].

Важный класс механизмов порождения языков образуют формальные грамматики (Formal Grammar), которых в 1959 году ввел впервые американский лингвист Хомский [24].

В формальной грамматике, порождающей язык ***L***, используются два непересекающихся множества символов:

1) конечное множество терминалов (terminals) – постоянных величин ***T***, из которых образуются цепочки языка ***L***;

2) непересекающееся с множеством ***T*** конечное множество нетерминалов (nonterminals) – переменных величин ***N***, которыми обозначают грамматические понятия, категории и т.п. языка ***L***.

3) Процесс порождения цепочек языка ***L*** описывается конечным множеством правил подстановки (rewriting rule) ***P***, каждое из которых состоит из пар цепочек (α, β). В такой паре первая компонента α является цепочкой, содержащей хотя бы один нетерминал, а вторая компонента может быть любой цепочкой, образованной из терминальных и/или нетерминальных символов. Она может быть и пустой цепочкой.

Соглашения I.31. Принимаются следующие соглашения :

(1) строчные латинские курсивные буквы a, b,…, z и арабские цифры 0, 1, ..., 9 обозначают терминалы;

(2) прописные латинские курсивные буквы A, B, …, X, Y, Z обозначают нетерминалы, при этом через S обозначается начальный нетерминальный символ;

(3) строчные греческие буквы α, β,..., ω обозначают цепочки, которые могут содержать как терминалы, так и нетерминалы, здесь ε – пустая цепочка;

(4) правило подстановки, являющееся парой цепочек (α, β) из множества $\boldsymbol{P}$, записывается в виде $\alpha \to \beta$;

(6) правила вида $\alpha \to \varepsilon$ называются ε (эпсилон)-правилами;

(7) эти соглашения распространяются также и на буквы с нижними и верхними индексами;

(8) правила вида $\alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_m \to \beta$ является сокращением m правил вида $\alpha_1 \to \beta$, $\alpha_2 \to \beta$, …, $\alpha_m \to \beta$ или:

$$\alpha_1 \to \beta$$
$$\alpha_2 \to \beta$$
$$\ldots$$
$$\alpha_m \to \beta$$

(9) правила вида $\alpha \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$ является сокращением n правил вида $\alpha \to \beta_1$, $\alpha \to \beta_2$, …, $\alpha \to \beta_n$ или:

$$\alpha \to \beta_1$$
$$\alpha \to \beta_2$$
$$\ldots$$
$$\alpha \to \beta_n$$

(10) правило вида $\alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_{m} \to \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$ является сокращением $\boldsymbol{m} \times \boldsymbol{n}$ правил, полученных из соглашении (6) и (7).

Определение I.3.1.1. *Формальной грамматикой* называется следующая четверка $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$, где:

$\boldsymbol{T}$ – непустое конечное множество терминальных символов (терминалов);

$\boldsymbol{N}$ – непустое конечное множество нетерминальных символов (нетерминалов), причем $\boldsymbol{T} \cap \boldsymbol{N} = \boldsymbol{\varnothing}$, $\boldsymbol{\varnothing}$ – пустое множество;

$\boldsymbol{P}$ – непустое конечное множество правил подстановок вида $\alpha \to \beta$, где $\alpha \in (\boldsymbol{T} \cup \boldsymbol{N})^* \times \boldsymbol{N} \times (\boldsymbol{T} \cup \boldsymbol{N})^*$, $\beta \in (\boldsymbol{T} \cup \boldsymbol{N})^*$, то есть,

$$\boldsymbol{P} \subseteq \{(\alpha,\beta): \alpha \in (\boldsymbol{T} \cup \boldsymbol{N})^* \times \boldsymbol{N} \times (\boldsymbol{T} \cup \boldsymbol{N})^* \& \beta \in (\boldsymbol{T} \cup \boldsymbol{N})^*\};$$

S – начальный нетерминал, $S \in \boldsymbol{N}$.

Правила вывода грамматики можно рассматривать как элементарные операции, которые, будучи применены в определенной последовательности к исходной цепочке, порождают лишь правильные цепочки. Сама последовательность правил, использованных в процессе порождения некоторой цепочки, является выводом этой цепочки.

Язык, определяемый грамматикой, является множеством конечных цепочек, которые состоят только из терминалов. Все эти терминальные цепочки выводятся, начиная с одной особой цепочки, состоящей только из одного начального нетерминала S.

Примеры I.3.1.1. Грамматика $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$ с параметрами: $\boldsymbol{P} = \{S \to E,\ E \to E+V|E-V|V,\ V \to V*F|V/F|F,\ F \to a|(E)\}$, $\boldsymbol{N} = \{S, E, V, F\}$, $\boldsymbol{T} = \{+, -, /, *, (,), a\}$ порождает скобочное алгебраическое выражение в инфиксной записи.

Для определения языка с помощью грамматики используются понятие *выводимой цепочки* и *отношение непосредственной выводимости*.

Определения I.3.1.2:

1. Пусть α,β,γ – выводимые цепочки грамматики $\boldsymbol{G}=<\boldsymbol{T},\boldsymbol{N},\boldsymbol{P},S>$. Тогда *выводимые цепочки* рекурсивно определяются так:

1) S – выводимая цепочка грамматики $\boldsymbol{G}$;

2) Если $\alpha\beta\gamma$ – выводимая цепочка грамматики $\boldsymbol{G}$ и в $\boldsymbol{P}$ имеется правило $\beta \to \sigma$, то $\alpha\sigma\gamma$ – тоже выводимая цепочка грамматики $\boldsymbol{G}$.

2. Выводимая цепочка грамматики $\boldsymbol{G}$, не содержащая нетерминальных символов из $\boldsymbol{N}$, называется *терминальной цепочкой*, порождаемой грамматикой $\boldsymbol{G}$.

3. Если $\alpha = \gamma\xi\delta$, $\beta = \gamma\eta\delta$ и $\alpha \to \beta$, $\xi \to \eta$ – правила вывода грамматики $\boldsymbol{G}$, то говорят, что между цепочками α и β установлено *отношение непосредственной выводимости*, которое означает, что в грамматике $\boldsymbol{G}$ цепочка β непосредственно выводится из цепочки α путем замены ξ на η и обозначается это отношение через $\alpha \Rightarrow_{\boldsymbol{G}} \beta$. Если грамматика

заранее известна, то показатель $\boldsymbol{G}$ в отношении непосредственной выводимости опускается и это отношение записывается как $\alpha \Rightarrow \beta$.

Запись вида $\alpha \Rightarrow^k \beta$ является k-той степенью отношения $\alpha \Rightarrow \beta$, если существуют $k+1$ цепочки $\alpha_0, \alpha_1, ..., \alpha_k$ такие, что $\alpha = \alpha_0$, $\alpha_k = \beta$ и $\alpha_{i-1} \Rightarrow \alpha_i$ $(1 \le i \le k)$. Эта последовательность цепочек называется *выводом длины k* цепочки β из цепочки α в грамматике $\boldsymbol{G}$.

Если существует $i \ge 1$ (или $i \ge 0$) выполняется отношение $\alpha \Rightarrow^i \beta$, то это записывается как $\alpha \Rightarrow^+ \beta$ (или $\alpha \Rightarrow^* \beta$). Здесь через $\Rightarrow^+$ обозначается транзитивное замыкания отношения $\Rightarrow$, а через $\Rightarrow^*$ – рефлексивное и транзитивное замыкания отношения $\Rightarrow$. При этом запись вида $\varphi \Rightarrow^+ \psi$ ($\varphi \Rightarrow^* \psi$) читается как: «$\psi$ *выводима из* φ *нетривиальным образом*» («ψ *выводима из* φ»).

Замечание I.3.1: $\alpha \Rightarrow^* \beta$ тогда и только тогда, когда $\alpha \Rightarrow^i \beta$ для некоторого $i \ge 0$, и $\alpha \Rightarrow^+ \beta$ тогда и только тогда, когда $\alpha \Rightarrow^i \beta$ для некоторого $i \ge 1$.

Определения 1.3.1.3:

1. Каждая цепочка, которая выводится из начального нетерминала грамматики, называется *сентенциальной формой*.

2. Выводимые цепочки, не содержащие нетерминальных символов, называются *терминальными цепочками*. Поэтому язык $\boldsymbol{L}(\boldsymbol{G})$ можно определить как множество терминальных цепочек, выводимых в грамматике $\boldsymbol{G}$.

3. *Я*зыком $\boldsymbol{L}(\boldsymbol{G})$, порождаемым грамматикой $\boldsymbol{G}$, называется множество терминальных цепочек, которые выводятся из одного начального нетерминала S посредством применения правило подстановки из множества $\boldsymbol{P}$, т.е. формально записывается как

$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{\tau: \ \tau \in \boldsymbol{T}^*, S \Rightarrow^* \tau\}.$$

Это означает, что любая принадлежащая языку $\boldsymbol{L}(\boldsymbol{G})$ цепочка является сентенциальной формой.

Примеры I.3.1.2.

1. Пусть задана грамматика $\boldsymbol{G_1}$ =<$\boldsymbol{T}$, $\boldsymbol{N}$, $\boldsymbol{P}$, S>, где $\boldsymbol{T}$ = {0,1} – множество терминалов, $\boldsymbol{N}$ = {A, S} – множество нетерминалов, $\boldsymbol{P}$ = {$S\to0A1$, $0A\to00A1$, $A\to\varepsilon$} – множество правило подстановки. Если рассмотрим вывод вида $S\Rightarrow0A1\Rightarrow00A11\Rightarrow0011$, то можно увидеть, что на первом шаге нетерминал S по правиле $S\to0A1$ заменяется на цепочку $0A1$, на втором шаге цепочка $0A$ по правиле $0A\to00A1$ заменяется на цепочку $00A11$, а на третьем шаге нетерминал A по правиле $A\to\varepsilon$ заменяется на пустую цепочку ε. Таким образом, можно говорить, что $S\Rightarrow^{3}0011$, $S\Rightarrow^{+}0011$, $S\Rightarrow^{*}0011$ и цепочка 0011 принадлежит языку $\boldsymbol{L}(\boldsymbol{G_1}) = \{0^n1^n:\ n>1\}$.

2. Грамматика с правилами $\boldsymbol{P_1}$ = {$S\to01S$, $S\to0$} и грамматика с правилами $\boldsymbol{P_2}$ = {$S\to0A$, $A\to10A$, $A\to\varepsilon$} эквивалентны.

3. Две грамматики для порождения алгебраичесих выражений, образованных операндами i, n и операциями +, * с одинаковыми терминальными символами $\boldsymbol{T}$ = {i, n, (,), +, *} и нетерминальными символами $\boldsymbol{N}$ = {S, F, H}, но с разными правилами:

$\boldsymbol{P_1}$ = {$S\to S+F$, $S\to F$, $F\to F*H$, $F\to H$,$F\to H$, $H\to i$, $H\to n$, $H\to(S)$};

$\boldsymbol{P_2}$ = {$S\to S+F|F$, $S\to S+F|S*F|F$, $F\to F*H|H$, $H\to i|n|(S)$};

эквивалентны.

Задания 1.3.1:

1. Построить все сентенциальные формы для грамматики с правилами:

$S\to A+B|B+A$, $A\to a$, $B\to b$.

2. Построить вывод заданной цепочки $a\text{-}b*a+b$ для грамматики с правилами:

$S\to K|F+S|K\text{-}S$, $K\to F|F*K$, $F\to a|b$.

3. Построить вывод заданной цепочки $aaabbbccc$ для грамматики с правилами:

$S\to aSBC|abC$, $CB\to BC$, $bB\to bb$, $bC\to bc$, $cC\to cc$.

4. Описать язык, порождаемый грамматикой

$S\to FF$, $F\to aFb$, $F\to ab$.

5. Описать язык, порождаемый грамматикой

$S \to Sc$, $S \to A$, $A \to aAb$, $A \to \varepsilon$.

6. Описать язык, порождаемый грамматикой

$S \to \varepsilon$, $S \to a$, $S \to b$, $S \to aSa$, $S \to bSb$.

7. Описать язык, порождаемый грамматикой

$S \to SA$, $SAA \to ASb$, $ASA \to b$, $A \to a$.

8. Описать язык, порождаемый грамматикой

$S \to aSA$, $S \to abc$, $bA \to bbc$, $cA \to Aa$.

9. Описать язык, порождаемый грамматикой

$S \to aAS$, $S \to B$, $Aa \to aaA$, $AB \to B$, $B \to a$.

Вопросы 1.3.1.

1. Эквивалентны ли следующие грамматики

$$S \to ab,\ S \to aKSb,\ K \to bSb,\ KS \to b,\ K \to \varepsilon$$

и

$$S \to aAb,\ A \to \varepsilon,\ A \to b,\ A \to S,\ A \to bSbS\ ?$$

2. Эквивалентны ли следующие грамматики

$$S \to aD,\ D \to bba,\ D \to baDa,\ D \to aDaDa$$

и

$$S \to aaE,\ S \to abD,\ E \to bDD,\ D \to aaEa,\ D \to abDa,\ D \to ba\ ?$$

3. Какому классу принадлежит грамматика

$$S \to abba,\ S \to baa\ ?$$

4. Какому классу принадлежит грамматика

$$S \to AD,\ A \to aA,\ A \to \varepsilon,\ D \to bDc,\ D \to \varepsilon$$

5. Эквивалентны ли грамматика с правилами

$$S \to AB,\ A \to a|Aa,\ A \to a|Aa$$

и грамматика с правилами

$$S \to AS|SB|AB,\ A \to a,\ B \to b\ ?$$

6. Эквивалентны ли грамматика с правилами

$$S \to cE,\ E \to ddc,\ E \to dcEc,\ E \to cEcEc$$

и грамматика с правилами

$$S \to ccA,\ S \to cdB,\ A \to dBB,\ B \to ccAc,\ B \to cdBc,\ B \to dc?$$

7. Как можно описать однозначной грамматикой язык, который порожден неоднозначной грамматикой $E \to E+E|E*E|(E)|i$.

I.3.2. Классы грамматик и их алгоритмические проблемы

В этом параграфе будут расммотрены классы грамматик и языков, рассмотрены их алгоритмические проблемы, а также будут предложены примеры, даны задания, сформулированы вопросы [1-9,11-18,21,25-32].

Определения I.3.2.1. С помощью формальных грамматик можно порождать различные классы языков, накладывая ограничения на их правила вывода:

1. Любая грамматика, правилам вывода которой не накладывается никакое ограничение принадлежит классу 0 и называется *неограниченной грамматикой* (НГ), – *unrestricted grammar* а множество цепочек, порождаемое этой грамматикой, будет *рекурсивно-перечислимым языком*.

2. Грамматика, в которой всем правилам вывода вида $\alpha \to \beta$ накладывается ограничение $\alpha = \xi H \zeta$, $\beta = \xi \eta \zeta$, $\xi \in (\boldsymbol{T} \cup \boldsymbol{N})^*$, $H \in \boldsymbol{N}$, $\eta \in (\boldsymbol{T} \cup \boldsymbol{N})^+$, $\zeta \in (\boldsymbol{T} \cup \boldsymbol{N})^*$ принадлежит классу 1 и называется *контекстно-зависимой* грамматикой – *context-sensitive grammar*, а множество цепочек, порождаемое этой грамматикой, будет *контекстно-зависимым языком*.

3. Грамматика, в которой всем правилам вывода вида $A \to \alpha$ накладывается ограничение $A \in \boldsymbol{N}$, $\alpha \in (\boldsymbol{T} \cup \boldsymbol{N})^*$, является класса 2 и называется *контекстно-свободной грамматикой – context-free grammar* и множество цепочек, порожденное этой грамматикой будет *бесконтекстным* (*контекстно-свободным*) *языком*.

4. Грамматика, в которой все правила вывода имеют вид $A \to \alpha B \beta$ или $A \to \alpha$, где A, $B \in \boldsymbol{N}$, $\alpha, \beta \in \boldsymbol{T}^*$, является класса 3 и называется *линейной грамматикой – linear grammar* (ЛГ). В линейной грамматике, если $\beta = \varepsilon$, то она будет *праволинейной грамматикой – right-linear grammar*, а если $\alpha = \varepsilon$, то она будет *леволинейной грамматикой – left-linear grammar*. Множество цепочек, порожденное *леволинейной грамматикой* будет называться *леволинейным языком*, а

множество цепочек, порожденное *праволинейной грамматикой* – *праволинейным языком*.

Замечания I.3.2:

1. В некоторых источниках контекстно-свободную грамматику называют бесконтекстной грамматикой (БГ), контекстно-зависимую грамматику называют контекстной грамматикой (КГ) или неукорочивающей грамматикой или грамматикой непосредственно составляющих.
2. Каждая линейная грамматика является безконтекстной грамматикой.
3. Каждая безконтекстная грамматика является контекстной грамматикой.
4. Каждая контекстная грамматика является грамматикой без ограничения.
5. Любой линейный язык является собственным подмножеством безконтекстного языка, но безконтекстный язык может не быть линейным.
6. Любой бесконтекстный язык, не содержащий пустую цепочку, будет собственным подмножеством контекстного языка.
7. Любой контекстный язык содержится в рекурсивно-перечислимом языке.
8. Если L_0, L_1, L_2, L_3 – языки, порождаемые грамматиками типа 0, 1, 2, 3 соответственно, то справедливо $L_3 \subseteq L_2 \subseteq L_1 \subseteq L_0$.

Примеры I.3.2.1:

1. Язык $\{a^n b^n c^n,\ n > 0\}$, порожденный грамматикой с правилами вывода $S \to aSBC$, $S \to aBC$, $CB \to BC$, $aB \to ab$, $bB \to bb$, $bC \to bc$, $cC \to cc$ будет рекурсивно-перечислимым. Здесь можно применять правила в любом порядке, только нужно обязательно применить правило $S \to aBC$ (для фиксации n), а правило $bC \to bc$ применять только после того, как справа от C не останется ни одного B (иначе этот B не сможет замениться на b и вывод не завершится терминальной цепочкой).

2. Множество булевских формул, заданное переменными a, b, c, будет бесконтекстным языком, так как оно порождается бесконтекстной грамматикой $\boldsymbol{P} = \{S \to \neg S,\ S \to S \wedge F,\ S \to S \vee F,\ S \to F,\ F \to a|b|c,\ F \to (S)\}$, $\boldsymbol{N}=\{S, F\}$, $\boldsymbol{T}=\{a, b, c, \neg, \vee, \wedge, (,)\}$.

3. Пусть $\boldsymbol{G} = <\{S\}, \{a, b\}, \{S \to aSa,\ S \to b\}, S>$. Тогда $aSa \Rightarrow^3 aaaaSaaaa$.

4. Язык $\{a^{2n-1},\ n \geq 1\}$, порожденный грамматикой вида $\boldsymbol{N}=\{S\}$; $\boldsymbol{T}=\{a\}$, $\boldsymbol{P} = \{S \to a,\ S \to aaS\}$ будет праволинейным языком.

5. Язык $\{a^{2n-1}\}$, порожденный грамматикой вида $\boldsymbol{N}=\{S\}$, $\boldsymbol{T}=\{a\}$, $\boldsymbol{P}=\{S \to a,\ S \to Saa\}$ будет леволинейным языком.

6. Грамматика с правилами $S \to aSa$, $S \to Q$, $Q \to bQ$, $Q \to \varepsilon$ является линейной, но не праволинейной, так как первое правило не имеет требуемого вида.

7. Пусть задан алфавит $\boldsymbol{T} = \{t_1, t_2, \ldots, t_n\}$. Тогда язык $\boldsymbol{T}^*$ является праволинейным, так как он порождается грамматикой с правилами:

$$S \to \varepsilon,\ S \to t_1 S,\ S \to t_2 S,\ \ldots,\ S \to t_n S.$$

8. Грамматика с правилами $S \to QQ$, $Q \to cQQ$, $Q \to bQ$, $S \to a$ является бесконтекстной, но не линейной, так как первые два правила не имеют требуемого вида.

9. Грамматика со следующими правилами $S \to QS$, $S \to US$, $S \to b$, $Qb \to Ab$, $A \to a$, $QA \to AAQ$, $UAb \to b$, $UAAA \to AAU$ не является бесконтекстной, т.к. последние 3 правила не имеют требуемого вида.

10. Грамматика с такими правилами $S \to ASQA$, $S \to AbA$, $A \to a$, $bQ \to bb$, $AQ \to UQ$, $UQ \to UV$, $UV \to QV$, $QV \to QA$ является контекстной, но не бесконтекстной, так как последние пять правил не имеют требуемого вида.

Определения I.3.2.2:

1. Если язык $\boldsymbol{L}(\boldsymbol{G})$, порождаемый грамматикой $\boldsymbol{G}$, не содержит ни одной конечной цепочки (конечного слова) из терминальных символов, то он называется *пустым языком*, т.е. $\boldsymbol{L}(\boldsymbol{G}) = \varnothing$.

2. Для того, чтобы язык $\boldsymbol{L}(\boldsymbol{G})$ не был пустым, должно быть хотя бы одно правило вида $\xi \to \omega$ и должен существовать вывод $S \Rightarrow^* \xi$, где $S \in N$ – начальный нетерминал, $\xi \in (\boldsymbol{T} \cup \boldsymbol{N})^* \times \boldsymbol{N} \times (\boldsymbol{T} \cup \boldsymbol{N})^*$, $\omega \in \boldsymbol{T}^*$.

3. Если в грамматике $\boldsymbol{G}$ правила вывода образуют замкнутый цикл, то такая грамматика порождает *бесконечный язык*, т.е. $\boldsymbol{L}(\boldsymbol{G})=\infty$;

4. Если для любой цепочки ξ и заданной грамматики $\boldsymbol{G}$ выполняется $\xi\in\boldsymbol{L}(\boldsymbol{G})$, то ξ является цепочкой языка $\boldsymbol{L}(\boldsymbol{G})$;

5. Если для любых двух грамматик $\boldsymbol{G'}$ и $\boldsymbol{G''}$ выполняется $\boldsymbol{L}(\boldsymbol{G'})=\boldsymbol{L}(\boldsymbol{G''})$, то грамматики $\boldsymbol{G'}$ и $\boldsymbol{G''}$ *эквивалентны*. Иначе говоря, две грамматики $\boldsymbol{G'}$ и $\boldsymbol{G''}$ эквивалентны, если они порождают один и тот же язык $\boldsymbol{L}(\boldsymbol{G'})=\boldsymbol{L}(\boldsymbol{G''})$.

Примеры I.3.2.2:

1. Грамматика с правилами $S\to Q$, $U\to abba$ порождает пустой язык, обозначаемый как $\varnothing$.

2. Грамматика с правилами $S\to aS$ порождает бесконечный язык, обозначаемый как ∞;

3. Грамматика с правилами $S\to abS$, $S\to a$ и грамматика с правилами $S\to aU$, $U\to baU$, $U\to\varepsilon$ являются эквивалентными.

Учитывая вышесказанные можно рассмотреть следующие алгоритмические проблемы грамматик:

1. *Проблема пустоты* – для заданной грамматики $\boldsymbol{G}$ выяснить будет ли $\boldsymbol{L}(\boldsymbol{G})$ пустым языком, т.е. $\boldsymbol{L}(\boldsymbol{G})=\varnothing$?

2. *Проблема принадлежности* – для любой цепочки ξ выяснить принадлежит ли она языку $\boldsymbol{L}(\boldsymbol{G})$, порожденному заданной грамматикой $\boldsymbol{G}$, т.е. $\xi\in\boldsymbol{L}(\boldsymbol{G})$?

3. *Проблема эквивалентности* – для любых двух грамматик $\boldsymbol{G'}$ и $\boldsymbol{G''}$ выяснить будут ли они эквивалентными, т.е. $\boldsymbol{L}(\boldsymbol{G'})=\boldsymbol{L}(\boldsymbol{G''})$?

4. *Проблема замкнутности* – при применении множественной операции к языкам определенного типа выяснитть будет ли результат иметь такой же тип?

5. *Проблема бесконечности* – для заданной грамматики $\boldsymbol{G}$ выяснить будет ли $\boldsymbol{L}(\boldsymbol{G})$ бесконечным языком, т.е. $\boldsymbol{L}(\boldsymbol{G})=\infty$?

Задания I.3.2:

1. Найдите линейную грамматику, порождающий следующий язык $\{a^m b^n c:\ \tau\in\{a,b\}^*, m\geq 0, |\tau|_b=2\}$.

2. Найдите линейную грамматику, порождающий следующий язык $\{a^n\tau:\ n\geq 1, m\geq 1$.

3. Найдите линейную грамматику, порождающий следующий язык $\{a,b\}^* - a^n b^n c^n:\ n\geq 0$.

4. Найдите линейную грамматику, порождающий следующий язык $\{\varphi a\psi\xi b:\ \varphi\in\{a,b\}^*, \psi\in\{a,b\}^*$.

5. Найдите линейную грамматику, порождающий следующий язык $\{a,b,c\}^* - \{\tau c\tau:\ \tau\in\{a,b\}^*\}$.

6. Найти праволинейную грамматику, эквивалентную грамматике $S\to KbbaK$, $K\to Ka$, $K\to Kb$, $S\to\varepsilon$.

7. Найти праволинейную грамматику, эквивалентную грамматике $S\to aSb$, $S\to K$, $S\to J$, $K\to aK$, $J\to Jb$, $K\to a$, $J\to\varepsilon$.

Вопросы I.3.2:

1. Существует ли такая праволинейная грамматика $\boldsymbol{G}$, что язык $\boldsymbol{L}(\boldsymbol{G})^R$ не порождается ни одной праволинейной грамматикой, имеющей столько же правил, сколько грамматика $\boldsymbol{G}$?

2. К какому типу относится грамматики с правилами:

   a) $S\to a|Ba,\ B\to Bb|b$;

   b) $S\to Ab,\ A\to Aa|ba$;

   c) $S\to 0A1|01,\ 0A\to 00A1,\ A\to 01$;

   d) $S\to AB,\ AB\to BA,\ A\to a,\ B\to b$;

   e) $S\to aC,\ S\to bD,\ C\to\varepsilon,\ D\to\varepsilon$?

3. Эквивалентны ли грамматики с правилами:

   $S\to AB,\ A\to a|Aa,\ B\to b|Bb$ и $S\to AS|SB|AB,\ A\to a,\ B\to b$?

4. Какой тип языка, порождаемого грамматикой с правилами:

   1) $S\to CD,\ C\to aCA,\ C\to bCB,\ AD\to aD,\ BD\to bD,\ Aa\to aA,\ Ab\to bA,\ Ba\to aB,\ Bb\to bB,\ C\to\varepsilon,\ D\to\varepsilon$?

   2) $S\to AB,\ A\to Aa|bB,\ B\to a|Sb$?

   3) $S\to aSBC|abC,\ CB\to BC,\ bB\to bb,\ bC\to bc,\ cC\to cc$?

   4) $S\to aS,\ S\to aA,\ A\to bA,\ A\to bZ,\ Z\to\varepsilon$?

   5) $S\to Ab,\ A\to Ab,\ A\to Za,\ Z\to\varepsilon$?

6) $S \rightarrow aB, B \rightarrow aB, B \rightarrow b$?

7) $S \rightarrow AB, A \rightarrow a, A \rightarrow ac, B \rightarrow b, B \rightarrow cb$?

I.3.3. Механизмы распознавания языков

В этом параграфе будут рассмотрены неформальное определения автомата и описываются его состав, конфигурация и работа, а также будут предложены примеры, даны задания, сформулированы вопросы [1-13,15-21,23-32].

Обычно под словом «автомат» понимается устройство, после включения самостоятельно выполняющее ряд заданных операций. Однако мы имеем дело с абстрактным автоматом, применяемым как математическая модель любых цифровых (дискретных) устройств, в которых все сигналы квантованы по уровню, а все действия квантованы по времени.

Абстрактный автомат (далее - автомат) может распознавать некоторое множество или преобразует одно множество в другое и состоит из ленты, головки и управляюшего устройства, а также может иметь и рабочую память.

Лента – линейная последовательность ячеек, каждая из которых может хранить только один символ из некоторого конечного входного (выходного) алфавита.

Лента автомата является потенциально бесконечной, при этом в каждый момент времени его работы занятыми считается конечное число ячеек. Граничные слева и справа от занятой области ячейки могут занимать специальные маркеры для обозначения начала и конца ленты. Маркер может стоять только на одном конце ленты или может отсутствовать вообще.

Входная (*рабочая*) *головка* – устройство, которое в каждый момент времени может обозревать только одну ячейку ленты. Головка может сдвинуться на одну ячейку влево или сдвинуться на одну ячейку вправо, либо остаться неподвижной на месте. Обычно предполагается, что головка только читает, т.е. во время работы автомата символы на ленте не меняются. Но можно рассматривать и такие распознаватели, головка которых и читает и пишет. Поэтому головка может быть читающей и пишущей.

Рабочая память – вспомогательное хранилище информации, предназначенное для чтения и записи данных. Рабочая память может быть организована в виде динамической структуры данных (очереди или стека).

Управляющее устройство – устройство, которое управляет поведением автомата и имеет конечную внутреннюю память для хранения конечного множества состояний. Он управляет поведением автомата с помощью функции (отношении), описывающего, как меняются состояния в зависимости от текущего состояния и текущего входного символа, читаемого головкой, и текущей информации, извлеченной с рабочей памяти, если она имеется. Управляющее устройство также определяет направление сдвига рабочей головки и то, какую информацию записать в рабочую память.

Автомат определяется заданием конечного множества состояний управляющего устройства, конечного множества допустимых входных символов, начального состояния и множества заключительных состояний, а также функции переходов, которая по текущему состоянию и текущему входному символу, являюшимися ее аргументами, указывает все возможные следующие состояния, являющиеся значениями этой функции. Работу автомата удобно описывать с помощью его конфигурации. Конфигурация автомата включает в себя:

– состояние управляющего устройства;

– содержимое входной ленты с положением входной головки;

– содержимое рабочей памяти вместе с положением рабочей головки, если она есть;

– содержимое выходной ленты, если она есть.

Конфигурация автомата может быть начальной, текущей и заключительной.

В *начальной конфигурации* внутренняя память содержит заранее записанный символ, обозначающий начальное состояние управляющего устройства; управляющее устройство находится в начальном состоянии; головка читает самый левый входной символ на

ленте; если имеется рабочая память, то она имеет заранее установленное начальное содержимое.

В *текущей конфигурации* внутренняя память содержит заранее записанные символы текущих состояний управляющего устройства; управляющее устройство находится в одном из текущих состояний; головка читает ни самый левый и ни самый правый текущий входной символ ленты; если имеется рабочая память, то она имеет заранее установленное текущее содержимое.

В *заключительной конфигурации* внутренняя память содержит заранее записанные символы, обозначающие заключительные состояния управляющего устройства; управляющее устройство находится в одном из заключительных состояний; головка обозревает правый концевой маркер или если маркер отсутствует, сошла с конца входной ленты; если имеется рабочая память, то она удовлетворяет некоторым условиям.

Перед началом работы автомат находится в начальной конфигурации, т.е. во внутреннюю память записывается символ обозначения начального состояния управляющего устройства, во входную ленту – входная цепочка, и если предусмотрена рабочей памяти, то в нее – соответствующую информацию.

Автомат работает по программе, состоящей из конечной последовательности *тактов*. Каждый такт строится из текущей (начальной) и следующей (заключительной) конфигураций.

В начале такта в памяти читается символ текущего состояния управляющего устройства, во входной ленте читается текущий входной символ и исследуется информация в рабочей памяти, если она есть. Затем, в зависимости от текущего состояния и прочитанной информации, определяются действия автомата:

(1) входная головка сдвигается вправо, влево или стоит на месте;

(2) записывается новый символ в текущую ячейку входной ленты или прежний символ не изменяется;

(3) в рабочую память записывается некоторая информация, если она есть;

(4) записывается символ на выходную ленту, если она есть.

(5) управляющее устройство переходит в другое состояние и номер (символ) этого состояние записывается во внутренюю память.

Итак, за один такт работы автомата входная головка может сдвинуться на одну ячейку влево, вправо или остаться на месте. В ходе работы автомата содержимое ячеек входной ленты не меняются, а содержимое ячеек выходной ленты и рабочей ленты могут изменяться.

Если автомат просматривает входную цепочку, выполняя последовательность тактов, которая начинается с начальной конфигурации и заканчивается заключительной конфигурацией, то он распознает эту цепочку.

Языком, распознаваемым автоматом называется множество цепочек, распознаваемых этим автоматом.

Примеры I.3.3:

1. Таксофон служит примером реализации автомата: распознает введенную монету и переходит в состояние набора номера.
2. Банкомат может служит автоматом: распознает введенную карточку и переходит в состояние ввода пинкода.
3. Турникет в метро является автоматом: разпознает введенный жетон и переходит в состояние открытие входа.

Задания I.3.3:

1. Опишите структуру входной ленты автомата.
2. Опишите функцию головки автомата.
3. Опишите функцию рабочей памяти автомата.
4. Опишите алгоритм работы автомата.
5. Опишете как рапознается цепочка.

Вопросы I.3.3:

1. Что содержит входная лента?
2. Чем определяется направление сдвига головки?
3. Из чего состоит конфигурация автомата?
4. Какие виды конфигурации имеются?
5. Из чего состоит распознаваемый автоматом язык?

I.3.4. Типы автоматов и их алгоритмические проблемы

В этом параграфе будут рассмотрены типы автоматов в зависимости от их состава, способа фиксации и определения состояний, функциональных возможностей и других параметров, будут обсуждены их алгоритмические проблемы, а также будут предложены примеры, даны задания, сформулированы вопросы [1-13,15-21,23-32].

В зависимости от направления движения головки автоматы разделяются на следующие два типа:

1) *Односторонний автомат* – автомат, у которого головка двигается только в одну сторону, например слева направо;

2) *Двухсторонний автомат* – автомат, у которого головка передвигается и влево и вправо.

В зависимости от фиксированности начального состояния автоматы разделяются на два типа:

1) *Инициальный автомат* – автомат, у которого в начальный момент состояние управляющего устройства может быть фиксированным.

2) *Неинициальный автомат* – автомат, у которого в начальный момент состояние управляющего устройства может быть произвольным.

В зависимости от способа определения состояний для следующего такта, автоматы разделяются на следующие два типа:

1) *Детерминированный автомат* (ДА) – автомат, у которого для каждого такта (кроме заключительго) следующее состояние управляющего устройства определяется однозначно, т.е. если для каждой его конфигурации будет существовать только один следующий такт;

2) *Недетерминированный автомат* (НА) – автомат, у которого для каждого такта (кроме заключительго) следующее состояние управляющего устройства определяется неоднозначно, т.е. если

для каждой его конфигурации будет существовать конечное множество возможных следующих альтернатиных тактов.

В зависимости от функциональной возможности автоматы разделяются на следующие два типа:

1) *распознаватели* - автоматы без выхода, которые имеют только функцию переходов из одного состояния в другое и распознают, принадлежит ли входная цепочка τ заданному языку $\boldsymbol{L}$, т.е. отвечает на вопрос: $\tau \in \boldsymbol{L}$**?**

2) *преобразователи* - автоматы с выходом, которые имеют наряду с функцией переходов и функцию преобразования входной цепочки τ в выходную цепочку γ при условии, что $\tau \in \boldsymbol{L}$.

В зависимости от сложности рабочей памяти автоматы разделяются на следующие типы автоматов:

1) *конечный автомат* (КА) – автомат, у которого отсутствует рабочая память.

2) *стековый автомат* (СА) – автомат, у которого рабочая память организована по принципу стека: последним вошел, первым вышел - Last Input, First Output (LIFO);

3) *линейно-ограниченный автомат* (ЛОА) – автомат, у которого рабочая память представляет собой линейную функцию длины входной цепочки;

4) *машина Тьюринга* (МТ) – автомат, у которого лента неограниченная в обе стороны (слева и справа).

Замечание I.3.4:

1. Недетерминированность автомата означает, что его управляющее устройство переходит во все свои следующие состояния, если угодно, дублируя себя так, что в каждом из возможных следующих состояний находится один экземпляр этого автомата.

2. Недетерминированность не следует путать со "случайностью", при которой можно случайно выбрать одно из следующих состояний с фиксированными вероятностями.

Недтерминированный автомат – удобная математическая абстракция, но, к сожалению, его очень сложно моделировать на практике (реализовать программно или аппаратно).

Между формальными грамматиками и автоматами существуют такая связь:

Языки класса 0 (неограниченные); ЛМТ, НГ в практике не используются.

Языки класса 1 (контекстно-зависимы); ЛОА, КЗГ используются для построения программ – переводчиков с естественных языков.

Языки класса 2 (контекстно-свободные); СА, КСГ используются для построения синтаксических анализаторов языков программирования.

Языки класса 3 (регулярные); КА, ЛГ используются для построения текстовых редакторов и отладчиков программ.

Аналогично формальным грамматикам, автоматы тоже имеют свои алгоритмические проблемы. К ним относятся:

1. *Проблема пустоты* – для заданного автомата $\boldsymbol{M}$ выяснить $\boldsymbol{L}(\boldsymbol{M})$ будет ли пустым языком, т.е. $\boldsymbol{L}(\boldsymbol{M}) = \varnothing$?

2. *Проблема принадлежности* – для любой цепочки ξ выяснить принадлежит ли она языку $\boldsymbol{L}(\boldsymbol{M})$, распознанному заданным автоматом $\boldsymbol{M}$, т.е. $\xi \in \boldsymbol{L}(\boldsymbol{M})$?

3. *Проблема эквивалентности* – для любых двух автоматов $\boldsymbol{M_i}$ и $\boldsymbol{M_j}$ ($\boldsymbol{i \neq j}$) выяснить будут ли они эквивалентными, т.е. $\boldsymbol{L}(\boldsymbol{M_i}) = \boldsymbol{L}(\boldsymbol{M_j})$?

4. *Проблема замкнутости* – при применении множественной операции к языкам определенного типа выяснить будет ли результат иметь такой же тип языка?

5. *Проблема бесконечности* – для заданного автомата $\boldsymbol{M}$ выяснить $\boldsymbol{L}(\boldsymbol{M})$ будет ли бесконечным языком?

6. *Проблема остановки* – для заданного автомата $\boldsymbol{M}$ и заданных входных данных выяснить завершится ли когда-либо работа $\boldsymbol{M}$ с этими данными.

Задания I.3.4:

1. Укажите разницу между односторонними и двусторонними автоматами.
2. Укажите разницу между инициальными и неинициальными автоматами.
3. Укажите разницу между детерминированными и недетерминированными автоматами.
4. Укажите автоматы, соответствующие регулярным выражениям и праволинейным грамматикам.
5. Укажите автоматы, соответствующие контекстно-свободным грамматикам.
6. Укажите автоматы, соответствующие контекстно-зависимым грамматикам.
7. Укажите автоматы, соответствующие грамматикам общего вида.

Вопросы I.3.4:

1. Какой класс языков распознает конечный автомат?
2. Какой класс языков распознает стековый автомат?
3. Какой класс языков распознает линейно-ограниченный автомат?
4. Какой класс языков распознает Машина Тьюринга?

5. Как формулируется проблема принадлежности в автоматах?
6. Как формулируется проблема эквивалентности в автоматах?
7. Как формулируется проблема остановки в автоматах?

I.3.5. Формальное определение автомата

В этом параграфе будут рассмотрены формальные определения автоматов без выходов, их конфигурации, такты работ и распознаваемые ими языки, обсуждены алгоритмические проблемы, а также будут предложены примеры, даны задания, сформулированы вопросы [1-5,12,13,16,21, 23,25-32].

Определение I.3.5.1. *Неинициальный автомат* ***A*** задается шестеркой вида $\boldsymbol{A} = <S, I, O, F_s, F_o, S_f>$, где:

$S = \{s_1, s_2, \ldots, s_K\}$ – множество состояний (алфавит состояний);

$I = \{i_1, i_2, \ldots, i_L\}$ – множество символов входной ленты (алфавит входной ленты);

$O = \{o_1, o_1, \ldots, o_M\}$ – множество символов выходной ленты (алфавит выходной ленты);

F_s: $S \times I \rightarrow \vartheta(S)$ – функция переходов, которая некоторым парам состояние–входной символ (s_k, i_l) ставит в соответствие состояние автомата $s_k = F_s(s_n, i_l)$; $s_k, s_n \in S$; $k \neq n$; $k=1, 2, \ldots, K$; $n=1, 2, \ldots, K$; $i_l \in I$; $l=1, 2, \ldots, L$; $\vartheta(S)$ – множество всех подмножеств множества S;

F_o: $S \times I \rightarrow O$ – функция выходов, которая некоторым парам состояние–входной символ (s_k, i_l) ставит в соответствие выходные символы $o_n = F_s(s_k, i_l)$; $s_k, s_n \in S$; $k \neq n$; $k=1, 2, \ldots, K$; $n=1, 2, \ldots, K$; $i_l \in I$; $l=1, 2, \ldots, L$; $o_m \in O$; $m=1, 2, \ldots, M$; $S_f \subseteq S$ – множество заключительных (финальных) состояний.

Определение I.3.5.2. *Инициальный автомат* ***A*** задается с выходной лентой семеркой вида $\boldsymbol{A} = <S, I, O, F_s, F_o, s_b, S_f>$, где параметры S, I, O, F_s, F_o, S_f – такие же как в неинициальном автомате; $s_b \in S$ – начальное состояние.

Обозначения выбраны по следующим соображениям: ***A*** – от «automaton», S / s – от «state», I / i – от «input», O / o – от «output», F – от «function», s_b – от «state begin», s_f – от «state final».

В дальнейшем мы не будем различать неинициальных и инициальных автоматов, так как по контексту будет видно о каком автомате идет речь.

Автомат называется *частично определенным*, если область определения функции F_s и F_o является подмножеством множества $S\times I$, т.е. $D(F_s)\subseteq S\times I$, $D(F_o)\subseteq S\times I$ и называется *полностью определенным*, если $D(F_s)= D(F_s)=S\times I$.

В дальнейшем мы будем рассматривать только автоматов распознавателей, работающие только в соответствии с функциями переходов.

Замечание I.3.5. Иногда вместо функции переходов (преобразования) удобно использовать отношение переходов (преобразования), которое принимает логическое значение «истина» или «ложь» .

Для перехода от функции переходов (преобразования) к отношению переходов (преобразования) нужно принять значение этой функции в качестве дополнительного аргумента отношения.

Формальное определение конфигурации без рабочей памяти будет пара $(s, \omega)\in S\times I^*$, где $s\in S$ и $\omega\in I^*$, I^* - множество всех возможных входных цепочек конечной длины.

Конфигурация автомата будет *начальной* (*заключительной*), если ее первый элемент в паре является начальным (заключительным) состоянием s_b (s_f), а второй элемент – входной (пустой) цепочкой.

Такт автомата – это бинарное отношение, определенное между двумя конфигурациями, которое обозначется как $\models$.

Наличие отношения $(s_t, i\omega) \models (s_{t+1}, \omega)$ означает, что если автомат находится в состоянии s_t и головка обозревает входной символ i, то за один такт автомат перейдет в состояние s_{t+1} (или в другое отличное от s_{t+1} состояние) и головка сдвинется на ячейку вправо. $\models^n$ является n–той степенью отношения $\models$: если существует цепочка (s_0,τ_0), (s_1,τ_1), (s_2,τ_2),…, (s_{n-1},τ_{n-1}), (s_n,τ_n) такая, что $s_0=s_b$, $\tau_0=\tau$, $s_k= s_f$, $\tau_k=\varepsilon$ *и* $\forall\ k,\ 1\le k\le n$ выполняются отношения $(s_{k-1},\tau_{k-1}) \models (s_k,\tau_k)$, где ε – пустая цепочка.

Если для любого $k\ge1$ (или $k\ge0$) выполняется отношение $(s_0,\tau) \models^k(s_k,\varepsilon)$, то это записывается как $(s_0,\tau) \models^+(s_k,\varepsilon)$ (или $(s_0,\tau) \models^*(s_k,\varepsilon)$),

где $\models^+$ – транзитивное замыкания, а $\models^*$ – рефлексивное и транзитовное замыкание отношения $\models$.

Входная цепочка распознается автоматом без выхода, если он выполняет последовательность конфигурацией, начиная с начальной конфигурации, в которой входная цепочка записана на ленте, может проделать последовательность шагов, заканчивающуюся заключительной конфигурацией.

Языком, распознаваемым автоматом, называется множество распознанных этим автоматом цепочек.

Пусть (s_b, τ) – начальная конфигурация автомата, (s_f, ε) – заключительная конфигурация автомата, тогда говорят, что:

1) автомат-распознаватель $\boldsymbol{A}$ распознает входную цепочку $\tau \in I^*$, если выполняется отношение $(s_b, \tau) \models^* (s_f, \varepsilon)$;

2) язык $\boldsymbol{L(A)}$ распознается автоматом-распознавателем $\boldsymbol{A}$, если он определяется как $\boldsymbol{L(A)} = \{\tau: \ \tau \in I^* \ \& \ (s_b, \tau) \models^* (s_f, \varepsilon)\}$;

При наличии команды автомат функционирует: если на такте t управляющее устройство находится в состоянии s и обозревает во входной ленте символ k, то оно записывает на выходную ленту символ o и переходит к следующему такту t+1, т.е. команды автомата можно записать в виде двух функций $o(t) = F_s(s(t), k(t))$, $s(t+1) = F_o(s(t), k(t))$.

Иногда команды автомата можно представить в виде троек:

$$<s(t), k(t), o(t)>, \quad <s(t), k(t), s(t+1)>$$

Примеры I.3.5. Для КА с параметрами $\boldsymbol{Q}$={1,2}; $\boldsymbol{T}$={*a*,*b*}; $\boldsymbol{I}$={1}; $\boldsymbol{F}$={2} и с командами Δ=<1, *aaa*, 1>, <1, *ab*, 2>, <1,*b*, 2>, <2, ε, 1>} диаграмма представлена на рисунке *I*.3.5, в котором веса ребер записаны регулярными выражения

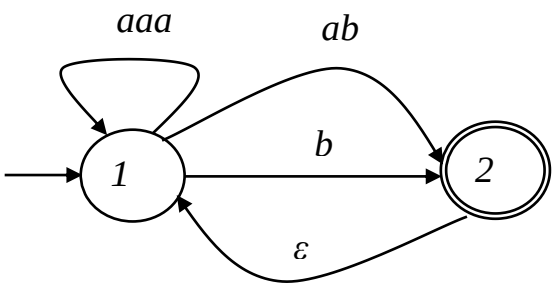


Рисунок I.3.5. Диаграмма конечного автомата.

Задания I.3.5:

1. Дайте формальное определение неинициального автомата.
2. Дайте формальное определение инициального автомата.
3. Построить конечный автомат, распознающий язык

$$\{a, b\}^*.$$

4. Построить конечный автомат, распознающий язык

$$\{\tau \in \{a, b, c\}^*: |\tau| \neq 1\}.$$

5. Построить конечный автомат, распознающий язык

$$\{a,b\}^* \setminus (\{a^n: \ n \geq 0\} \cup \{b^n: n \geq 0\}).$$

6. Построить конечный автомат, распознающий язык

$$\{a\omega b: \ \omega \in \{a, b\}^*\} \cup \{b\omega a: \ \omega \in \{a, b\}^*\}.$$

7. Построить конечный автомат, распознающий множество всевозможных цепочек, состоящих из a и b, начинающихся с a.
8. Построить конечный автомат, распознающий множество всевозможных цепочек, состоящих из a и b, заканчивающихся на b.
9. Построить конечный автомат, распознающий множество всевозможных цепочек, состоящих из a, b, c, начинающихся с a и заканчивающихся на c.
10. Построить конечный автомат, распознающий множество всевозможных цепочек, состоящих из латинских маленьких букв и арабских цифр.

Вопросы I.3.5:

1. Из чего состоит конфигурация автомата?
2. Какие виды конфигурации имеются?
3. Из чего состоит такт автомата?
4. Как задается функция переходов автомата?
5. Как задается функция выходов автомата?
6. Когда автомат считается частично определенным?
7. Когда автомат считается полностью определенным?
8. Что определяет множество распознанных автоматом цепочек?
9. Как записывается команда конечного автомата?
10. Как определяется инициальный автомат?
11. Что такое диаграмма автомата?

II. РЕГУЛЯРНЫЕ ЯЗЫКИ

II.1. Механизмы порождения регулярных языков

II.1.1. Регулярные множества и регулярные выражения

В этой части будут рассмотрены способы определения регулярных языков (регулярные выражения, регулярные уравнения, системы регулярных уравнений, линейные грамматики, относящиеся к механизмам порождения регулярных языков; конечные автоматы, являющиеся механизмами распознавания регулярных языков), обсуждены свойства регулярных языков и эквивалентность различных механизмов определения этих языков, а также будут предложены примеры, даны задания, сформулированы вопросы [1-3,16-21,24,25,28-32].

Пусть $\varnothing$ – пустое множество, $\{\varepsilon\}$ – множество пустых цепочек $\boldsymbol{T}$ – конечный алфавит и t – символ из $\boldsymbol{T}$, т.е. $t \in \boldsymbol{T}$. Тогда можно дать следующее определение.

Определение II.1.1.1 *Регулярное множество* в алфавите $\boldsymbol{T}$ определяется рекурсивно следующим образом:

Базис рекурсии:

(1) $\varnothing$ – регулярное множество в алфавите $\boldsymbol{T}$;

(2) $\{\varepsilon\}$ – регулярное множество в алфавите $\boldsymbol{T}$;

(3) $\{t\}$ – регулярное множество в алфавите $\boldsymbol{T}$;

Рекурсивное расширение: Если $\boldsymbol{P}$ и $\boldsymbol{Q}$ – регулярные множества в алфавите $\boldsymbol{T}$, то регулярными являются:

(4) $\boldsymbol{P} \cdot \boldsymbol{Q}$ – конкатенация множеств $\boldsymbol{P}$ и $\boldsymbol{Q}$;

(5) $\boldsymbol{P} \cup \boldsymbol{Q}$ – объединение множеств $\boldsymbol{P}$ и $\boldsymbol{Q}$;

(6) $\boldsymbol{P}^*$ – итерация множества $\boldsymbol{P}$;

Рекурсивное заключение:

Регулярное множество в алфавите $\boldsymbol{T}$ определяются только правилами (1) – (6).

Итак, множество в алфавите $\boldsymbol{T}$ регулярно тогда и только тогда, когда оно либо $\emptyset$, либо $\{\varepsilon\}$, либо $\{t\}$ для некоторого $t \in \boldsymbol{T}$, либо его можно получить из этих множеств применением конечного числа операций объединения, конкатенации и итерации.

Определение II.1.1.2. Регулярные выражения, описывающие регулярные множества в конечном алфавите $\boldsymbol{T}$, определяются рекурсивно следующим образом:

Базис рекурсии:

(1) ϕ – регулярное выражение, задающее регулярное множество $\emptyset$ в алфавите $\boldsymbol{T}$;

(2) ε – регулярное выражение, задающее регулярное множество $\{\varepsilon\}$ в алфавите $\boldsymbol{T}$,

(3) если $t \in \boldsymbol{T}$, то t – регулярное выражение, задающее регулярное множество $\{t\}$ в алфавите $\boldsymbol{T}$;

Рекурсивное расширение:

Если p и q – регулярные выражения, обозначающие регулярные множества $\boldsymbol{P}$ и $\boldsymbol{Q}$ в алфавите $\boldsymbol{T}$ соответственно, то:

(4) $p \cdot q$ – регулярное выражение, задающее $\boldsymbol{P} \cdot \boldsymbol{Q}$ в алфавите $\boldsymbol{T}$;

(5) $p \vee q$ – регулярное выражение, задающее $\boldsymbol{P} \cup \boldsymbol{Q}$ в алфавите $\boldsymbol{T}$;

(6) p^* – регулярное выражение, задающее $\boldsymbol{P}^*$ в алфавите $\boldsymbol{T}$;

Рекурсивное заключение:

Регулярные выражения определяются только правилами (1) – (6).

Для построения регулярного выражения использовались операция конкатенация $\cdot$, операция дизъюнкция $\vee$ и операция итерация $*$. Наивысшим приоритетом обладает операция $*$, затем идет $\cdot$, а операция $\vee$ будет последним. Обычно в записи регулярного выражения знак операции «$\cdot$» опускается.

Мы будем пользоваться записью p^+ для сокращенного обозначения выражения pp^* и, кроме того, будем устранять из

регулярных выражений лишние скобки там, где это не может привести к недоразумениям. Так, $0 \vee 10^*$ означает $(0 \vee (1(0^*)))$.

Для каждого регулярного множества можно найти по крайней мере одно регулярное выражение, задающее это множество и обратно, для каждого регулярного выражения можно построить регулярное множество, обозначаемое этим выражением.

Отметить, что для каждого регулярного множества существует бесконечно много обозначающих его регулярных выражений.

Будем говорить, что два регулярных выражения равны, если они обозначают одно и то же множество. Для эквивалентного преобразования имеются алгебраические законы.

При практическом описании лексических структур бывает полезно сопоставлять регулярным выражениям некоторые имена, и ссылаться на них по этим именам. Для определения таких имен мы будем использовать запись вида

$$d_1 = r_1$$

$$d_2 = r_2$$

...

$$d_n = r_n$$

где d_i - различные имена, а каждое r_i - регулярное выражение над символами $\boldsymbol{T} \cup \{d_1, d_2, ..., d_{i-1}\}$, т.е. символами основного алфавита и ранее определенными символами-именами. Таким образом, для любого r_i можно построить регулярное выражение над $\boldsymbol{T}$, повторно заменяя имена регулярных выражений на обозначаемые ими регулярные выражения.

Примеры II.1.1:

1. Регулярное выражение $(a \vee b)^*$ определяет регулярное множество $\{a, b\}^*$.
2. Регулярное выражение $(0 \vee 1)^*011$ определяет регулярное множество всех цепочек, составленных из 0 и 1, оканчивающихся цепочкой 011.

3. Регулярное выражение $a(a \vee 0)^*$ определяет регулярное множество всех цепочек из $\{0, a\}^*$, начинающихся с a.

4. Регулярное выражение $(a \vee b)(a \vee b \vee 0 \vee 1)^*$ определяет регулярное множество всех цепочек из $\{0, 1, a, b\}^*$, начинающихся с a или b.

5. Регулярное выражение $(00 \vee 11)^* \cdot ((01 \vee 10) \cdot (00 \vee 11)^* \cdot (01 \vee 10) \cdot (00 \vee 11)^*)^*$ определяет регулярное множество всех цепочек нулей и единиц, содержащих четное число нулей и четное число единиц.

6. Регулярное выражение $a(\varepsilon \vee a) \vee b$ определяет регулярное множество $\{a, b, aa\}$.

7. Регулярное выражение $a(a \vee b)^*$ определяет множество всевозможных цепочек, состоящих из a и b, начинающихся с a.

8. Регулярное выражение $(a \vee b)^*(a \vee b)(a \vee b)^*$ определяет множество всех непустых цепочек, состоящих из a и b, т.е. множество $\{a, b\}^+$.

9. Регулярное выражение $((s \vee 1)(s \vee 1)(s \vee 1))^*$ определяет множество всех цепочек, состоящих из 0 и 1, длины которых делятся на 3.

10. Регулярное выражение ϕ^* определяет множество $\{\varepsilon\}$.

11. Регулярное выражение $\varepsilon \vee \phi$ определяет множество $\{\varepsilon\} \cup \varnothing$.

12. $(a \vee b) \cdot (a \vee b) = aa \vee ab \vee ba \vee bb$.

13. Для понятия идентификатор, состоящего из букв и цифр, и регулярное выражение для начинающего с буквы регулярное выражение строится так:

$L = a \vee b \vee c \vee d \vee e \vee f \vee g \vee h \vee i \vee j \vee k \vee l \vee m \vee n \vee o \vee p \vee q \vee s \vee t \vee u \vee v \vee w \vee x \vee y \vee z$;

$D = 0 \vee 1 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7 \vee 8 \vee 9$;

$I = L \cdot (L \vee D)^*$.

14. Пусть задано алфавит $\boldsymbol{T} = \{x, y\}$. Тогда согласно определению

$$\boldsymbol{L}((xy)^* \cdot (1 \vee x)) = \{(xy)^n : \ n \geq 0\} \cup \{(xy)^n x : \ n \geq 0\}.$$

15. Множества чисел в десятичной записи:

$Digit = 0 \vee 1 \vee 2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7 \vee 8 \vee 9$;

$Integer = Digit^+$;

$Fraction = Integer \vee \varepsilon$;

$Exponent = (E(+ \vee - \vee \varepsilon) Integer) \vee \varepsilon$.

Задания II.1.1:

1. Найти регулярное выражение, задающее множество $\{a, b\}^*$.
2. Найти регулярное выражение, задающее множество $\{a, bc^*\}^*$.
3. Найти регулярное выражение, задающее множество $\{ab, cd\}^*$.
4. Найти регулярное выражение, задающее множество $\{ab, b\}^*$.
5. Найти регулярное выражение, задающее множество $\{a^*, b^*\}^*$
6. Написать регулярное выражение, которое порождает предложение, состоящего из слов, пробелов и точек, и имеющее, следующую структуру: предложение начинается со слов, состоящих только из латинских маленьких букв; слова в предложении разделяется друг от друга пробелами; все предложение заканчивается единственной точкой.
7. Написать регулярное выражение, которое описывает структуру, включающую день, месяц, год и, представленную в виде "<цифра><цифра>.<цифра><цифра>.<цифра><цифра><цифра><цифра>".
8. Найти регулярное выражение для языка

$$\{\omega \in \{a, b\}^*: \ (|\omega|_a - |\omega|_b)...3\}.$$

9. Найти регулярное выражение для языка

$$\{\omega \in \{a, b\}^*: \ (|\omega|_a - |\omega|_b)...4\}.$$

10. Найти регулярное выражение для языка $\boldsymbol{L}_1 \cap \boldsymbol{L}_2 \cap \boldsymbol{L}_3$, где $\boldsymbol{L}_1 = (aaab \vee c \vee d)^*$, $\boldsymbol{L}_2 = (a^*ba^*ba^*bc \vee d)^*$, $\boldsymbol{L}_3 = ((a \vee b)^*c(a \vee b)^*cd)^*$.

Вопросы II.1.1:

1. Как определяется регулярное выражение в алфавите $\boldsymbol{T}$?
2. Какое множество определяет регулярное выражение $(ab)^+$?
3. Какое множество определяет регулярное выражение $(aa \vee bb)$?
4. Какое множество определяет регулярное выражение

$$a(\varepsilon \vee a) \vee b?$$

5. Какое множество определяет регулярное выражение

$$a(a \vee b)^*?$$

6. Какое множество определяет регулярное выражение ε^*?
7. Какое множество определяет регулярное выражение

$$((a \vee bc)^* \cdot a)?$$

8. Какое множество определяет регулярное выражение

$(c \cdot (ab \vee cd)^*)$?

9. Какое множество определяет регулярное выражение ϕ?

II.1.2. Регулярная алгебра и регулярные уравнения

В этом параграфе будут рассмотрены регулярная алгебра и регулярные уравнения, а также будут предложены примеры, даны задания, сформулированы вопросы [1,16-21,25,29,32,32].

Обычно под алгеброй понимают множество заданных элементов, над которыми определены некоторые операции, и установлены свойства этих операций. В случае регулярной алгебры, заданными элементами являются цепочки в алфавите ***T***, над которыми определены операции ∨, · и *, а свойства этих операций устанавливаются следующей леммой.

Лемма II.1.2.1. Если α, β и γ – регулярные выражения, отличные от ϕ и ε, тогда:

(1) $\phi\cdot\alpha=\alpha\cdot\phi=\phi$

(2) $\varepsilon\cdot\alpha=\alpha\cdot\varepsilon=\alpha$

(3) $\alpha\cdot\beta\neq\beta\cdot\alpha$

(4) $\phi\vee\alpha=\alpha\vee\phi=\alpha$

(5) $\alpha\vee\alpha=\alpha$

(6) $\alpha\vee\beta=\beta\vee\alpha$

(7) $\alpha\cdot(\beta\cdot\gamma)=(\alpha\cdot\beta)\cdot\gamma=\alpha\cdot\beta\cdot\gamma$

(8) $\alpha\vee(\beta\vee\gamma)=(\alpha\vee\beta)\vee\gamma=\alpha\vee\beta\vee\gamma$

(9) $\alpha\cdot(\beta\vee\gamma)=\alpha\cdot\beta\vee\alpha\cdot\gamma$

(10) $(\alpha\vee\beta)\cdot\gamma=\alpha\cdot\gamma\vee\beta\cdot\gamma$

(11) $\varepsilon^*=\varepsilon$

(12) $\phi^*=\varepsilon$

(13) $\alpha^*=\alpha\vee\alpha^*$

(14) $(\alpha^*)^*=\alpha^*$

Доказательство. Пусть регулярные выражения α, β и γ задают множества A, B и $\boldsymbol{G}$ соответственно. Тогда:

(1) регулярное выражение $\phi\cdot\alpha$ определяет регулярное множество $\varnothing\cdot\boldsymbol{A}$, а регулярлярное выражение $\alpha\cdot\phi$ определяет регулярное множество $\boldsymbol{A}\cdot\varnothing$. По свойству операции $\cdot$ над множествами получим $\varnothing\cdot\boldsymbol{A} = \boldsymbol{A}\cdot\varnothing = \varnothing$, следовательно, $\phi\cdot\alpha = \alpha\cdot\phi = \phi$.

(2) регулярлярное выражение $\varepsilon\cdot\alpha$ определяет регулярное множество $\{\varepsilon\}\cdot\boldsymbol{A}$, а регулярлярное выражение $\alpha\cdot\varepsilon$ определяет регулярное множество $\boldsymbol{A}\cdot\{\varepsilon\}$. По свойству операции $\cdot$ над множествами получим $\{\varepsilon\}\cdot\boldsymbol{A}=\boldsymbol{A}\cdot\{\varepsilon\}=\boldsymbol{A}$, следовательно, $\varepsilon\cdot\alpha=\alpha\cdot\varepsilon=\alpha$.

(3) регулярное выражение $\alpha\cdot\beta$ определяет регулярное множество $\boldsymbol{A}\cdot\boldsymbol{B}$, а регулярное выражение $\beta\cdot\alpha$ определяет регулярное множество $\boldsymbol{B}\cdot\boldsymbol{A}$. По свойству операции $\cdot$ над множествами получим $\boldsymbol{A}\cdot\boldsymbol{B} \neq \boldsymbol{B}\cdot\boldsymbol{A}$, следовательно, $\alpha\cdot\beta \neq \beta\cdot\alpha$.

(4) регулярное выражение $\phi\vee\alpha$ определяет регулярное множество $\varnothing\cup\boldsymbol{A}$, а регулярное выражение $\alpha\vee\phi$ определяет регулярное множество $\boldsymbol{A}\cup\varnothing$. По определению операции объединения над множествами получим равенство $\varnothing\cup\boldsymbol{A} = \boldsymbol{A}\cup\varnothing = \boldsymbol{A}$, следовательно, $\phi\vee\alpha = \alpha\vee\phi = \alpha$.

(5) регулярное выражение $\alpha\vee\alpha$ определяет регулярное множество $\boldsymbol{A}\cup\boldsymbol{A}$, а по определению операции объединения над множествами получим $\boldsymbol{A}\cup\boldsymbol{A}=\boldsymbol{A}$, следовательно, $\alpha\vee\alpha=\alpha$.

(6) регулярное выражение $\alpha\vee\beta$ определяет регулярное множество $\boldsymbol{A}\cup\boldsymbol{B}$, а регулярное выражение $\beta\vee\alpha$ определяет регулярное множество $\boldsymbol{B}\cup\boldsymbol{A}$. Но $\boldsymbol{A}\cup\boldsymbol{B} = \boldsymbol{B}\cup\boldsymbol{A}$, по определению объединения над множествами, следовательно, $\alpha\vee\beta = \beta\vee\alpha$.

(7) регулярное выражение $\alpha\cdot(\beta\cdot\gamma)$ определяет регулярное множество $\boldsymbol{A}\cdot(\boldsymbol{B}\cdot\boldsymbol{G})$, а регулярное выражение $(\alpha\cdot\beta)\cdot\gamma$определяет регулярное множество $(\boldsymbol{A}\cdot\boldsymbol{B})\cdot\boldsymbol{G}$. Из теории множеств получим следующие равенства $\boldsymbol{A}\cdot(\boldsymbol{B}\cdot\boldsymbol{G}) = (\boldsymbol{A}\cdot\boldsymbol{B})\cdot\boldsymbol{G} = \boldsymbol{A}\cdot\boldsymbol{B}\cdot\boldsymbol{G}$, следовательно, $\alpha\cdot(\beta\cdot\gamma) = (\alpha\cdot\beta)\cdot\gamma = \alpha\cdot\beta\cdot\gamma$.

(8) регулярное выражение $\alpha\vee(\beta\vee\gamma)$ определяет регулярное множество $\boldsymbol{A}\cup(\boldsymbol{B}\cup\boldsymbol{G})$, а регулярное выражение $(\alpha\vee\beta)\vee\gamma$ определяет

регулярное множество $(\boldsymbol{A}\cup\boldsymbol{B})\cup\boldsymbol{G}$. Из теории множеств известно, что $\boldsymbol{A}\cup(\boldsymbol{B}\cup\boldsymbol{G}) = (\boldsymbol{A}\cup\boldsymbol{B})\cup\boldsymbol{G} = \boldsymbol{A}\cup\boldsymbol{B}\cup\boldsymbol{G}$, следовательно имеем выражение вида $\alpha\vee(\beta\vee\gamma) = (\alpha\vee\beta)\vee\gamma = \alpha\vee\beta\vee\gamma$.

(9) регулярное выражение $\alpha\cdot(\beta\vee\gamma)$ определяет $\boldsymbol{A}\cdot(\boldsymbol{B}\cup\boldsymbol{G})$, а регулярное выражение $\alpha\cdot\beta\vee\alpha\cdot\gamma$ определяет регулярное множество $\boldsymbol{A}\cdot\boldsymbol{B}\cup\boldsymbol{A}\cdot\boldsymbol{G}$. Из теории множеств известно, что $\boldsymbol{A}\cdot(\boldsymbol{B}\cup\boldsymbol{G}) = \boldsymbol{A}\cdot\boldsymbol{B}\cup\boldsymbol{A}\cdot\boldsymbol{G}$, следовательно, $\alpha\cdot(\beta\vee\gamma) = \alpha\cdot\beta\vee\alpha\cdot\gamma$.

(10) регулярное выражение $(\alpha\vee\beta)\cdot\gamma$ определяет регулярное множество $(\boldsymbol{A}\cup\boldsymbol{B})\cdot\boldsymbol{G}$, а регулярное выражение $\alpha\cdot\gamma\vee\beta\cdot\gamma$ определяет регулярное множество $\boldsymbol{A}\cdot\boldsymbol{G}\cup\boldsymbol{B}\cdot\boldsymbol{G}$. Из теории множеств известно, что $(\boldsymbol{A}\cup\boldsymbol{B})\cdot\boldsymbol{G} = \boldsymbol{A}\cdot\boldsymbol{G}\cup\boldsymbol{B}\cdot\boldsymbol{G}$, следовательно, $(\alpha\vee\beta)\cdot\gamma = \alpha\cdot\gamma\vee\beta\cdot\gamma$.

(11) регулярное выражение ε^* определяет регулярное множество $\{\varepsilon\}^*$, а в свою очередь справедливо равенство $\{\varepsilon\}^*=\{\varepsilon\}\cup\{\varepsilon\}\cup\{\varepsilon\}\cdot\{\varepsilon\}\cup\{\varepsilon\}\cdot\{\varepsilon\}\cdot\{\varepsilon\}\cup\ldots=\{\varepsilon\}$, т.е. $\{\varepsilon\}^*=\{\varepsilon\}$. А множежество $\{\varepsilon\}$ описывается регулярным выражением ε, поэтому $\varepsilon^* = \varepsilon$.

(12) Для любых α справедливо $\alpha^* = \varepsilon\vee\alpha\vee\alpha\cdot\alpha\vee\alpha\cdot\alpha\cdot\alpha\vee\ldots$. Если здесь заменим все α на ϕ, применим несколько раз, сначала свойства (1), затем (5), а в самом конце свойство (4) получим $\phi^* = \varepsilon$.

(13) если в равенстве $\alpha^* = \varepsilon \vee \alpha \vee \alpha\cdot\alpha \vee \alpha\cdot\alpha\cdot\alpha \vee\ldots$ с обеих сторон прибавим α с помощью $\vee$ и к его правой части применим один раз свойство (5), тогда получим $\alpha \vee \alpha^*=\alpha^*$, т.е. $\alpha^* = \alpha \vee \alpha^*$.

(14) После раскрутки операции итерации внутри скобки получим $(\alpha^*)^*= (\varepsilon\vee\alpha\vee\alpha\cdot\alpha\vee\alpha\cdot\alpha\cdot\alpha\vee\ldots)^*$. Теперь раскрутим внешнюю итерацию и применим несколько раз свойства (2), (4), (5), (11) и (13) можно получить $(\alpha^*)^* = \alpha^*$.

Используя эти свойства регулярных выражений можно упрощать сложные регулярные выражения. Например, регулярное выражение $(0\vee(1(0^*)))$ упрощается на равнозначное выражение $0\vee10^*$.

В дальнейшем мы не будем различать регулярное выражение и обозначаемое им множество, если это не приводит к недоразумениям.

Например, в силу этого соглашения символ a будет представлять множество $\{a\}$.

Примеры II.1.2.1.

$\emptyset^* = \varepsilon$, так как $\emptyset^* = \varepsilon \vee \emptyset \vee \emptyset\emptyset \vee \emptyset\emptyset\emptyset \ldots = \varepsilon \vee \emptyset = \varepsilon \Rightarrow \emptyset^* = \varepsilon$.

Теперь рассмотрим уравнения, которые получаются из регулярных выражений.

Определение I.1.2.1. *Регулярными уравнениями* называются уравнения, коэффициентами и неизвестными которых служат регулярные множества.

Рассмотрим, например, регулярное уравнение

$$X = \alpha X \vee \beta, \qquad (1)$$

где X регулярное множество в некотором алфавите, а коэфициенты α и β – регулярные выражения в том же алфавите.

Теорема I.1.2.1:

1) Если в уравнении (1) коэфициент α не содержит пустую цепочку, т.е. $\varepsilon \notin \alpha$, то уравнение имеет единственное решение, равное регулярному выражению $\alpha^*\beta$.

2) Если в уравнении (1) коэфициент α содержит пустую цепочку, т.е. $\varepsilon \in \alpha$, то уравнение имеет решение $\alpha^*(\beta \vee \gamma)$.

Доказательство:

1) Если $\varepsilon \notin \alpha$, то $\forall i$ и $\forall X$ верно $\alpha^i\beta \subset X$, т.е. $\alpha^*\beta \subset X$. Пусть имеется $\xi \in X$, $\xi \notin \alpha^*\beta$, ξ – самая короткая; $\xi = \xi_\alpha \xi'$, где $\xi_\alpha \in \alpha$. Тогда $\xi_\alpha \notin \varepsilon$ и ξ' короче, чем ξ. Поэтому $\xi' \in \alpha^*\beta \Rightarrow \xi \in \alpha^*\beta \Rightarrow X = \alpha^*\beta$.

Теперь поставив регулярное выражение $\alpha^*\beta$ в правую часть уравнения вместо переменной X можно получить:

$$\alpha\alpha^*\beta \vee \beta = \alpha^+\beta \vee \beta = (\alpha^+ \vee \varepsilon)\,\beta = \alpha^*\beta.$$

Из них выходит решение уравнения (1) в виде $X = \alpha^*\beta$.

2) Пусть, $\varepsilon \in \alpha$ (т.е. $\alpha\alpha^* = \alpha^*$) и $\alpha^*(\beta \vee \gamma)$ является решением уравнения (1), где γ – произвольный, не обязательно регулярный язык. Отсюда $\alpha^*(\beta \vee \gamma)$ соответствует уравнению

$$\alpha\alpha^*(\beta \vee \gamma) \vee \beta = \alpha^+\beta \vee \alpha^*\gamma \vee \beta = (\alpha^+ \vee \varepsilon)\beta \vee \alpha^*\gamma = \alpha^*\beta \vee \alpha^*\gamma = \alpha^*(\beta \vee \gamma).$$

Таким образом, доказано, что решением уравнения (1) будет $X = \alpha^*(\beta \vee \gamma)$. Это показывает, что уравнение имеет бесконечно много решений. В этом случае нужно взять минимальное решение уравнения, называемое минимальной неподвижной точкой, им будет множество $X = \alpha^*\beta$.

Примеры II.1.2.2. Требуется решить уравнения:

1. $X = 01 \vee 1X$.

Решение:

$X=01 \vee 1X$

$X=1X \vee 01$

$X=(1)^*01$

$01 \vee (1)(1)^*=01 \vee 1^+=(1^+ \vee \varepsilon)\ 01=(1)^*01$

2. $A=X^*BA \vee B^*X$

Решение:

$A=(X^*A)^*A^*X$

$A=(X^*A)(X^*A)^*A^*X \vee A^*X$

$A=(X^*A)^+ A^*X \vee A^*X$

$A=((X^*A)^+ \vee E)\ A^*X$

$A=(X^*A)^*\ A^*X$

3. $A = 0A \vee (\varepsilon \vee b)$

Решение:

$A = 0^*\ (\varepsilon \vee 1)$

$00^*(\varepsilon \vee 1) \vee (\varepsilon \vee 1) =$

$(00^* \vee \varepsilon)\ (\varepsilon \vee 1) =$

$(0^+ \vee \varepsilon)\ (\varepsilon \vee 1) =$

$0^*\ (\varepsilon \vee 1) =$

$0^*\varepsilon \vee 0^*1 =$

$0^*\ (\varepsilon \vee 1)$

Задания I.1.2:

1. Упростите регуляроное выражение $(a \vee b \vee ab)^*$.

2. Упростите регуляроное выражение $(a^*b)^* \vee (b^*a)^*$.
3. Упростите регуляроное выражение $(ba \vee a^*ab)^*$.
4. Упростите регуляроное выражение $(b^+a)^*b \vee 1)b^*$.
5. Найдите решения регулярного уравнения $Z = \varphi Z \vee \psi$.
6. Найдите решения регулярного уравнения $X = (a \vee \beta)X$.
7. Найдите решения регулярного уравнения $X = (a \vee \beta)X \vee \beta$.
8. Найдите решения регулярного уравнения $X = (a \vee \beta)^*X$.
9. Найдите решения регулярного уравнения $X = a\,X \vee \beta$.

Вопросы II.1.2:

1. Равны ли регулярные выражения $((a \vee b)^* \vee aa)^*$ и $(aa \vee b \vee ab)^*$?
2. Равны ли регулярные выражения $(ab \vee b)^*(a \vee b)$ и $(a \vee b)^*(a \vee b)^*$?
3. Равны ли регулярные выражения $(b \vee cd^*a)^*cd^*$ и $b^*c(d \vee ab^*c)^*$?
4. В каком случае решение регулярнного уравнения будет единственным?
5. В каком случае решение регулярнного уравнения будет много?
6. Каким методом находится решение регулярного уравнения?
7. Какие регулярные языки порождаются следующими регулярными выражениями (0*1*)0; (01*)0; (00∨11∨(01∨10)*(00∨11)?
8. Как можно упростить следующие регулярные выражения:

   $(00^*)0 \vee (00)^*$?

   $(0 \vee 1)(\varepsilon \vee 00)^+ \vee (0 \vee 1)$?

   $(0 \vee \varepsilon)0^*1$?

   $(a \vee b)(\varepsilon \vee ab)^+ \vee (b \vee a)$?

   $(aa \vee \varepsilon)(a \vee b)\,ab$?

   $a^*(a \vee b)\,)(\varepsilon \vee a)b$?

   $a(a \vee b)(a \vee b)b$?
9. Будут ли эквивалентными заданные регулярные выражения $(p^*q^*)^* = (q^*p^*)^*$?
10. Будут ли эквивалентными заданные регулярные выражения $p(qp)^*$ и $(pq)^*p$?
11. Будут ли эквивалентными заданные регулярные выражения $p^*(p \vee q)^*$, $(p \vee qp^*)^*$ и $(p \vee q)^*$?

II.1.3. Системы регулярных уравнений

В этом параграфе будут рассмотрены системы регулярных уравнений и их стандартный вид, а также будут предложены примеры, даны задания, сформулированы вопросы [1,16-21,25,29,32].

Рассмотрим систему с двумя уравнениями:

$$\begin{cases} X = \alpha_1 X \vee \alpha_2 Y \vee \alpha_3 \\ Y = \beta_1 X \vee \beta_2 Y \vee \beta_3 \end{cases} \qquad (1)$$

где α_i и β_i для всех i = 1,2,3 являются регулярными выражениями в алфавите ***T***.

Для ее решения нужно выполнить следующие шаги:

Шаг 1. Находится выражение $X = \alpha_1^*(\alpha_2 Y \vee \alpha_3)$.

Шаг 2. Находится выражение $Y = \beta_2^*(\beta_1 X \vee \beta_3)$.

Шаг 3. Поставив выражение из шага 1 в выражение из шага 2 можно получить $Y = \beta^*_2(\beta_1(\alpha_1^*(\alpha_2 Y \vee \alpha_3)) \vee \beta_3)$.

Шаг 4. Избавляясь от Y в выражении на шаге 3 можно получить выражение $Y = \beta^*_2(\beta_1(\alpha_1^*(\alpha_2^*\alpha_3)) \vee \beta_3)$.

Шаг 5. Выражение для Y из шага 4 подставив в выражение из шага 1 можно получить $X = \alpha_1^*(\alpha_2(\beta_2^*(\beta_1(\alpha_1^*(\alpha_2^*\alpha_3)) \vee \beta_3)) \vee \alpha_3)$.

Таким образом, решенияи системы уравнения будут:

$$X = \alpha_1^*(\alpha_2(\beta_2^*(\beta_1(\alpha_1^*(\alpha_2^*\alpha_3)) \vee \beta_3)) \vee \alpha_3)$$
$$Y = \beta^*_2(\beta_1(\alpha_1^*(\alpha_2^*\alpha_3)) \vee \beta_3)$$

Теперь можно рассматривать стандартную систему регулярных уравнений.

Определение I.1.3.1. Систему уравнений с регулярными коэффициентами назовем *стандартной* системой с множеством неизвестных $\Delta = \{X_1, X_2, \ldots, X_n\}$, если её вид:

$$\begin{cases} X_1 = \alpha_{10} \vee \alpha_{11} X_1 \vee \alpha_{12} X_2 \ldots \vee \alpha_{1n} X_n \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ X_n = \alpha_{n0} \vee \alpha_{n1} X_1 \vee \alpha_{n2} X_2 \ldots \vee \alpha_{nn} X_n \end{cases}$$

Здесь все коэффициенты α_{ij} – регулярные множества в алфавите $\boldsymbol{T}$, не пересекающемся с множеством Δ, $\Delta \cap \boldsymbol{T} \neq \emptyset$.

Коэффициентами уравнений являются выражения α_{ij}. Заметим, что если $\alpha_{ij} = \phi$ (такое регулярное выражение возможно), то в уравнении для неизвестной X_i нет члена, содержащего неизвестную X_j. Аналогично, если $\alpha_{ij}=\varepsilon$, то в уравнении для X_i член, содержащий X_j – это просто X_j. Иными словами, ϕ играет роль коэффициента 0, а ε – роль коэффициента 1 в обычных линейных уравнениях.

Теперь рассмотрим алгоритм решения стандартной системы уравнений с регулярными коэффициентами:

Вход. Стандартная система $\boldsymbol{Q}$ уравнений, коэффициенты которых являются регулярные выражения в алфавите $\boldsymbol{T}$, и с множеством неизвестных $\Delta = \{X_1, X_2, \ldots, X_n\}$.

Выход. Решение системы $\boldsymbol{Q}$ в виде $X_1 = \alpha_1, \ldots, X_n = \alpha_n$, где αi– регулярное выражение в алфавите $\boldsymbol{T}$.

Метод. Метод напоминает решение системы линейных уравнений методом исключения Гаусса.

Шаг 1. Положить i=1.

Шаг 2. Если $i = n$, перейти к шагу 4. В противном случае с помощью тождеств леммы 1 записать уравнение для X_i в виде $X_i=\alpha X_i \vee \beta$, где α–регулярное выражение в алфавите $\boldsymbol{T}$, а β – регулярное выражение вида $\beta_0 \vee \beta_i X_{i+1} \vee \ldots \vee \beta_n X_n$, причем все β_i – регулярные выражения в алфавите $\boldsymbol{T}$. (Мы увидим, что это всегда возможно.) Затем в правых частях уравнений для $X_{i+1}, \ldots, X_n$ заменить X_i регулярным выражением $\alpha^* \beta$.

Шаг 3. Увеличить i на 1 и вернуться к шагу 2.

Шаг 4. Записать уравнение для X_n в виде $X_n=\alpha X_n \vee \beta$, где α и β – регулярные выражения в алфавите $\boldsymbol{T}$. (После выполнения шага 2 для каждого $i<n$ в правой части уравнения для X_i не будет неизвестных $X_1,\ldots,X_{i-1}$. В частности, на шаге 4 этим свойством будет обладать и уравнение для X_n. Перейти к шагу 5 (при этом $i=n$).

Шаг 5. Уравнение для X_i, имеет вид $X_i = \alpha X_i \vee \beta$, где α и β – регулярные выражения в алфавите $\boldsymbol{T}$. На выходе будет $X_i = \alpha^*\beta$ и в уравнения для $X_{i-1},\ldots,X_1$ подставить $\alpha^*\beta$ вместо X_i.

Шаг 6. Если $i = 1$, остановиться. В противном случае уменьшить i на 1 и вернуться к шагу 5.

Определение I.1.3.2. Пусть $\boldsymbol{Q}$ – стандартная система уравнений с множеством неизвестных $\Delta = \{X_1, X_2,\ldots,X_n\}$ и с коэффициентами в алфавите $\boldsymbol{T}$. Отображение f множества Δ в множество языков в алфавите $\boldsymbol{T}$ называют *решением* системы, если после подстановки в каждое уравнение $f(X)$ вместо X для каждого $X\in\Delta$ уравнения становятся равенствами множеств.

Отображение f: $\Delta\rightarrow\vartheta(\boldsymbol{T}^*)$ называют *наименьшей неподвижной точкой* системы, если f – решение и для любого другого решения g справедливо $f(X) \subseteq g(X)$ для всех $X\in\Delta$, здесь $\vartheta(\boldsymbol{A}^*)$ – множество всех подмножеств множества $\boldsymbol{A}^*$.

Следующие две леммы содержат полезную информацию о наименьших неподвижных точках.

Лемма I.1.3.1. Каждая стандартная система уравнений $\boldsymbol{Q}$ с неизвестными Δ обладает единственной наименьшей неподвижной точкой.

Доказательство. Пусть $f(X) = \{\omega:\ \omega\in g(X)$ для всех решений g системы $\boldsymbol{Q}\}$ для всех $X\in\Delta$. Можно прямо показать, что f – решение и $f(X) \subseteq g(X)$ для всех решений g. Таким образом, f – единственная наименьшая неподвижная точка системы $\boldsymbol{Q}$.

Теперь дадим некоторую характеристику наименьшей неподвижной точки.

Лемма I.1.3.2. Пусть Q_1 и Q_2 – системы уравнений до и после одного применения шага 2 алгоритма 1. Тогда Q_1 и Q_2 имеют одну и ту же наименьшую неподвижную точку.

Лемма I.1.3.3. Пусть Q_1 и Q_2 – системы уравнений до и после одного применения шага 5 алгоритма 1. Тогда Q_1 и Q_2 имеют одну и ту же наименьшую неподвижную точку.

Теорема I.1.3.1. Алгоритм 1 находит наименьшую неподвижную точку стандартной системы уравнений.

Доказательство. После того как шаг 5 применен для всех i, все уравнения принимают вид $X_i = \alpha_i$, где α_i – регулярное выражение в алфавите T. Ясно, что отображение f для которого $f(X_i) = \alpha_i$, и будет наименьшей неподвижной точкой этой системы.

Примеры I.1.3. Решить систему регулярных уравнений:

1. $\begin{cases} X_1=\varepsilon\vee 01^*X_1\vee 11X_2 \\ X_2=00X_1\vee 1X_2\vee 1 \end{cases}$

Решение:

$X_1=(01^*)^*(\varepsilon\vee 11X_2)$

$X_2=00(01^*)^*(\varepsilon\vee 11X_2)\vee 1X_2\vee 1$

$X_2=(00(01^*)^*11\vee 1)^*(00(01^*)^*\vee 1$

$X_1=(01^*)^*(\varepsilon\vee 1100(01^*)^*11\vee 1)^*(00(01^*)^*\vee 1))$

2. $\begin{cases} X_1=aX_2\vee b\vee cX_1 \\ X_2=cX_1\vee cX_2\vee ab \end{cases}$

Решение:

$X_1=c^*(aX_2\vee b)$

$X_2=c(c^*(aX_2\vee b))\vee cX_2\vee ab$

$X_2=(cc^*a\vee c)X_2\vee cc^*b\vee ab$

$X_2=(cc^*a\vee c)^*(cc^*b\vee ab)$

$X_1=c^*(a(cc^*a\vee c)^*(cc^*b\vee ab)\vee b)$

3. $\begin{cases} X_1 = a \vee bX_2 \vee cX_1 \\ X_2 = c \vee bX_1 \vee aX_2 \end{cases}$

Решение:

$X_1 = c^*(a \vee bX_2)$

$X_2 = c \vee b(c^*(a \vee bX_2)) \vee aX_2$

$X_2 = c \vee bc^*a \vee (bc^*b \vee a)X_2$

$X_2 = (bc^*b \vee a)^*(c \vee bc^*a)$

$X_1 = c^*(a \vee b(bc^*b \vee a)^*(c \vee bc^*a))$

Задания I.1.3:

1. Найдите решения системы регулярного уравнений:

$$\begin{cases} X = \alpha_1 X \vee \beta_1 Y \\ Y = \alpha_2 X \vee \beta_2 Y \end{cases}$$

2. Найдите решения системы регулярного уравнений:

$$\begin{cases} X = \alpha_1 X \vee \beta_1 Y \vee \gamma_1 \\ Y = \alpha_2 X \vee \beta_2 Y \vee \gamma_2 \end{cases}$$

3. Приведите эту систему уравнений в стандартную:

$$\begin{cases} X = a_1 X \vee b_1 Y \vee c_1 \\ Y = a_2 X \vee b_2 Y \vee c_2 \end{cases}$$

4. Решите систему уравнений с регулярными коэфицентами:

$$\begin{cases} X_1 = abX_1 \vee b \vee aX_2 \\ X_2 = aaX_3 \vee bX_1 \\ X_3 = abX_2 \vee c \vee cX_3 \end{cases}$$

5. Решите систему уравнений с регулярными коэфицентами:

$$\begin{cases} X_1 = a_{10} \vee a_{11}X_1 \vee a_{12}X_2 \vee a_{13}X_3 \\ X_2 = a_{20} \vee a_{21}X_1 \vee a_{22}X_2 \vee a_{23}X_3 \\ X_3 = a_{30} \vee a_{31}X_1 \vee a_{32}X_2 \vee a_{33}X_3 \end{cases}$$

6. Решите систему уравнений с регулярными коэфицентами:

$$\begin{cases} X_1 = a_{10} \vee a_{11}X_1 \vee a_{12}X_2 \\ X_2 = a_{20} \vee a_{21}X_1 \vee a_{22}X_2 \end{cases}$$

7. Решите систему уравнений с регулярными коэфицентами:

$$\begin{cases} X_1 = 001^* X_1 \vee 1X_3 \vee 0X_2 \\ X_2 = 11X_2 \vee 0X_3 \vee 01X_1 \\ X_3 = 01X_3 \vee (0 \vee 1)X_1 \end{cases}$$

8. Решите систему уравнений с регулярными коэфицентами:

$$\begin{cases} X_1 = 00X_2 \vee 11X_2 \\ X_2 = 0X_3 \vee 01X_2 \\ X_3 = 1X_1 \vee 0X_3 \end{cases}$$

9. Найти решение стандартной системы регулярных уравнений:

$$\begin{cases} X_1 = a_{10} \vee a_{11}X_1 \vee a_{12}X_2 \vee a_{13}X_3 \\ X_2 = a_{20} \vee a_{21}X_1 \vee a_{22}X_2 \vee a_{23}X_3 \\ X_3 = a_{30} \vee a_{31}X_1 \vee a_{32}X_2 \vee a_{33}X_3 \end{cases}$$

10. Напишите алгоритм нахождения наименьшей неподвижной точки стандартной системы регулярных уравнений.

Вопросы I.1.3:

1. Что определяет решение системы регулярных уравнений?
2. Какие шаги нужно проделать, чтобы решить систему регулярных уравнений?

$$\begin{cases} X = (\alpha_1 \vee \alpha_2\beta_2^*\beta_1)^*(\alpha_3 \vee \alpha_2\beta_2^*\beta_3) \\ Y = (\beta_2 \vee \beta_1\alpha_1^*\alpha_2)^*(\beta_3 \vee \beta_1\alpha_1^*\alpha_3) \end{cases}$$

3. Какую систему регулярных уравнений называют стандартной?
4. Как определяется неподвижная точка системы регулярных уравнений?
5. Что такое минимальная неподвижная точка системы регулярных уравнений?
6. Имеет ли каждая система регулярных уравнений минимальную неподвижную точку?

7. В каком случае можно определить минимальную неподвижную точку системы регулярных уравнений?

II.1.4. Линейные грамматики

В этом параграфе будут рассмотрены линейные грамматики, их виды и свойства, а также предложены примеры, даны задания, сформулированы вопросы [1-9,11-13,15-21,22-25,28-32].

Пусть задана формальная грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N}$ – конечное множество нетерминалов, $\boldsymbol{T}$ – конечное множество терминалов, $\boldsymbol{T} \cap \boldsymbol{N} = \emptyset$, $\boldsymbol{P}$ – конечное множество правил вывода, S – *начальный нетерминал*, $S \in \boldsymbol{N}$. Тогда можно дать следующие определения:

Определения II.1.4:.

1. Если в грамматике $\boldsymbol{G}$ для каждого A, $B \in \boldsymbol{N}$, $\tau \in \boldsymbol{T}^*$ все правила вывода задаются в виде $A \to \tau B$ или $A \to \tau$, то она называется *праволинейной грамматикой*.

Множество цепочек, порожденное праволинейной грамматикой является *праволинейным языком*.

2. Если в грамматике $\boldsymbol{G}$ для каждого A, $B \in \boldsymbol{N}$, $\tau \in \boldsymbol{T}^*$ все правила вывода задаются в виде $A \to B\tau$ или $A \to \tau$, то она называется *леволинейной грамматикой*.

Множество цепочек, порожденное леволинейной грамматикой является *леаволинейным языком*.

3. Если в грамматике $\boldsymbol{G}$ для каждого $A \in \boldsymbol{N}$, $B \in \boldsymbol{N}$, $\alpha \in \boldsymbol{T}^*$, $\beta \in \boldsymbol{T}^*$ все правила вывода задаются в виде $A \to \alpha B \beta$ или $A \to \alpha$, то он называется *линейной грамматикой*.

Множество цепочек, порожденное линейной грамматикой является *линейным языком*.

В линейной грамматике в цепочке во время процесса вывода не будет лишнего нетерминала и если $\beta = \varepsilon$, то она будет праволинейной, а если $\alpha = \varepsilon$, то она будет леволинейной.

(1) Праволинейная грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ называется регулярной, если начальный нетерминал S не встречается в правой части никакого правила, т.е, каждое ее правило, кроме $S \to \varepsilon \in \boldsymbol{P}$, имеет вид либо $\boldsymbol{A} \to a\boldsymbol{B}$, либо $\boldsymbol{A} \to a$, где $\boldsymbol{A}, \boldsymbol{B} \in \boldsymbol{N}$, $a \in \boldsymbol{T}$.

4. Праволинейная грамматика находится $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ в *нормальной форме*, если в ней каждое правило имеет вид $A \to \varepsilon$, $A \to a$ или $A \to aB$, где $A \in \boldsymbol{N}$, $B \in \boldsymbol{N}$, $a \in \boldsymbol{T}$.

5. Леволинейная грамматика находится $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ в *нормальной форме*, если в ней каждое правило имеет вид $A \to \varepsilon$, $A \to a$ или $A \to Ba$, где $A \in \boldsymbol{N}$, $B \in \boldsymbol{N}$, $a \in \boldsymbol{T}$.

7. Линейная грамматика находится в нормальной форме, если в линейной грамматике каждое правило имеет вид $A \to \varepsilon$, $A \to \alpha$, $A \to \alpha B$ или $A \to B\alpha$, где $A \in \boldsymbol{N}$, $B \in \boldsymbol{N}$, $\alpha \in \boldsymbol{T}^*$.

Теперь можно установить свойства этих грамматик и языков с помощью следующих теорем:

Теорема II.1.4.1. Каждая праволинейная грамматика эквивалентна некоторой праволинейной грамматике в нормальной форме.

Теорема II.1.4.2. Если праволинейный язык не содержит пустого слова, то он порождается некоторой праволинейной грамматикой в нормальной форме без ε-правил.

Теорема II.1.4.3. Каждая праволинейная грамматика эквивалентна некоторой регулярной грамматике.

Теорема II.1.4.4. Каждая линейная грамматика эквивалентна некоторой линейной грамматике в нормальной форме.

Теорема II.1.4.5. Если линейный язык не содержит пустого слова, то он порождается некоторой линейной грамматикой в нормальной форме без ε-правил.

Теорема II.1.4.6. Язык $\boldsymbol{L}$ является линейным тогда и только тогда, когда язык $\boldsymbol{L}\backslash\{\varepsilon\}$ является линейным.

Теорема II.1.4.7. Пусть $\boldsymbol{L}$ - линейный язык над алфавитом $\boldsymbol{T}$. Тогда найдется такое положительное целое число k, что для любой цепочки $\lambda \in \boldsymbol{L}$ длины не меньше k можно подобрать цепочки $\alpha, \beta, \gamma, \delta, \tau \in \boldsymbol{T}^*$,

для которых верно $\alpha\beta\gamma\delta\tau = \omega$, $\beta\delta \neq \varepsilon$ (то есть $\beta \neq \varepsilon$ или $\delta \neq \varepsilon$), $|\alpha\beta|+|\delta\tau| \leq k$ и $\alpha\beta^i\gamma^i\delta\tau \in \boldsymbol{L}$ для всех $i \in \boldsymbol{N}$

Примеры II.1.4:

1. Рассмотрим грамматику $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N} = \{S, A\}$, $\boldsymbol{T} = \{a, b\}$, $\boldsymbol{P} = \{S \to aA, A \to aA, A \to b\}$. В этой грамматике порождается праволинейный язык $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n b : n = 1,2,...\}$ с помощью правил вывода $S \Rightarrow aA \Rightarrow aaA \Rightarrow aaaA \Rightarrow ... \Rightarrow a...aaab$.

2. Рассмотрим грамматику $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N} = \{S, A\}$, $\boldsymbol{T} = \{a, b\}$, $\boldsymbol{P} = \{S \to Aa, A \to Aa, A \to b\}$. В этой грамматике порождается леволинейный язык $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{ba^n : n = 1,2,...\}$ с помощью правил вывода $S \Rightarrow Aa \Rightarrow Aaa \Rightarrow Aaaa \Rightarrow ... \Rightarrow baaa...a$.

3. Рассмотрим праволинейный (леволинейный) язык $\{a^{2n-1}\}$, состоящий из цепочек вида a, aaa, $aaaaa$, ... Он порождается праволинейной (леволинейной) грамматикой $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, состоящей из множества $\boldsymbol{T} = \{a\}$, $\boldsymbol{N} = \{S\}$ и $\boldsymbol{P} = \{S \to a, S \to aaS\}$ ($\boldsymbol{P} = \{S \to a, S \to Saa\}$). По виду правил можно увидеть, что заданный язык будет праволинейным (леволинейным) языком.

4. Рассмотрим язык $\boldsymbol{L} = \{a^m b^m a^n b^n : m \geq 0, n \geq 0\}$ над алфавитом $\{a,b\}$. Утверждение теоремы II.1.4.7 не выполняется для любого натурального числа k. Следовательно, язык $\boldsymbol{L}$ не является линейным.

5. Язык $\{\omega \in \{a,b\}^* : |\omega|_a = 2, |\omega|_b = 2\}$ порождается линейной грамматикой. Заданный язык не содержит пустой цепочки, в любой цепочке число вхождения и a и b должно быть равно 2. Поэтому он порождается праволинейной грамматикой.

6. Пусть грамматика $\boldsymbol{G}$ имеет правила $S \to 0A|1S|\varepsilon$, $A \to 0B$, $B \to 0S$. Чтобы определить к какому классу относится язык $\boldsymbol{L}(\boldsymbol{G})$, построим альтернативные выводы $S \to 0A \to 00B \to 000S \to 0001S \to 0001\varepsilon \to 0001$ и $S \to 1S \to 10A \to 100B \to 1000S \to 1000e \to 1000$. В результате получим праволинейный язык $\boldsymbol{L}(\boldsymbol{G}) = \{(0^n 1) : n \geq 1\}$ тілін аламыз

Задания II.1.4:

1. Найти праволинейную грамматику, порождающую язык $\{\tau \in \{a,b\}^* : |\tau|_a \geq 2, |\tau|_b \geq 2\}$.

2. Найти праволинейную грамматику, эквивалентную грамматике $S \to E$, $S \to bE$, $S \to caE$, $E \to a$, $E \to bS$.

3. Найти праволинейную грамматику в нормальной форме без ε-правил, порождающую язык $\{a^k b^m c^n : k \geq 0, m \geq 1, n \geq 0\}$.

4. Найти праволинейную грамматику в нормальной форме без ε-правил, порождающую язык

$$\{a,b\}^* - (\{a^n : n \geq 0\} \square \{\{a^k b^m c^n : k \geq 0, m \geq 1, n \geq 0\}\ b^n : n \geq 0\}.$$

5. Найти линейную грамматику в нормальной форме без ε-правил, порождающую язык $\{a^n b^n c^m : n \geq 1, m \geq 1\}$.

6. Опишите язык, пораждаемый следующими правилами:

$$S \to 0A|1S|\varepsilon,\ A \to 0B|1A,\ B \to 0S|1B$$

7. Опишите язык, пораждаемый грамматикой:

$$\boldsymbol{T} = \{a, b, d\},\ \boldsymbol{N} = \{A, B, D\},\ S = A,$$

$$\boldsymbol{P} = \{A \to aB, B \to aB, B \to b, B \to bD, D \to d, D \to dD, A \to aD, A \to a\}$$

Вопросы II.1.4:

1. Имеются ли языки $\boldsymbol{L_1}$ и $\boldsymbol{L_2}$ такие, что $\boldsymbol{L_1}$ будет праволинейным и $\boldsymbol{L_2}$ будет леволинейным, а $\boldsymbol{L_1} \cup \boldsymbol{L_2}$ не будет линейным языком?

2. Имеются ли языки $\boldsymbol{L_1}$ и $\boldsymbol{L_2}$ такие, что $\boldsymbol{L_1}$ будет праволинейным и $\boldsymbol{L_2}$ будет леволинейным, а $\boldsymbol{L_1} \cap \boldsymbol{L_2}$ не будет линейным языком?

3. Существует ли такая праволинейная грамматика $\boldsymbol{G}$, что язык $\boldsymbol{L}(\boldsymbol{G})^R$ не порождается ни одной праволинейной грамматикой, имеющей столько же правил, сколько грамматика $\boldsymbol{G}$?

4. Существует ли праволинейная грамматика $\boldsymbol{G}$, что язык $\boldsymbol{L}(\boldsymbol{G})^R$ не порождается ни одной праволинейной грамматикой с количеством правил $n+1$ (где n - количество правил в $\boldsymbol{G}$)?

5. Существует ли праволинейная грамматика $\boldsymbol{G}$ с тремя нетерминалами, что язык $\boldsymbol{L}(\boldsymbol{G})^R$ не порождается ни одной праволинейной грамматикой с тремя нетерминалами?

6. Какому типу грамматики относятся следующие правила?

$$S \to 0A|1S|\varepsilon,\ A \to 0B|1A,\ B \to 0S|1B$$

7. Относят ли следующие правила к праволинейной грамматике?

$$S \to AB,\ A \to Aa|bB,\ B \to a|Sb$$

II.2. Распознающие механизмы регулярных языков

II.2.1. Недетерминированные и детерминированные конечные автоматы

В этой главе будут рассмотрены недетерминированные и детерминированные конечные автоматы, описаны способы определения регулярных языков с помощью конечных автоматов, а также будут предложены примеры, даны задания, сформулированы вопросы [1-3,5,10,13,19,23,25,26,32].

Конечные автоматы являются распознавателями регулярных языков. Сначала даются формальные определения недетерминированных и детерминированных конечных автоматов, затем описываются языки, распознаваемые ими, а в конце доказывается их эквивалентность.

Конечные автоматы являются одним из простейших и наиболее распространенными среди распознавателей. В составе конечного автомата имеются *конечная лента*, *внутренняя память*, *внешняя память*, *головка*, *управляющее устройство*. Структура конечного автомата на нижем рисунке *II*.2.1.1.

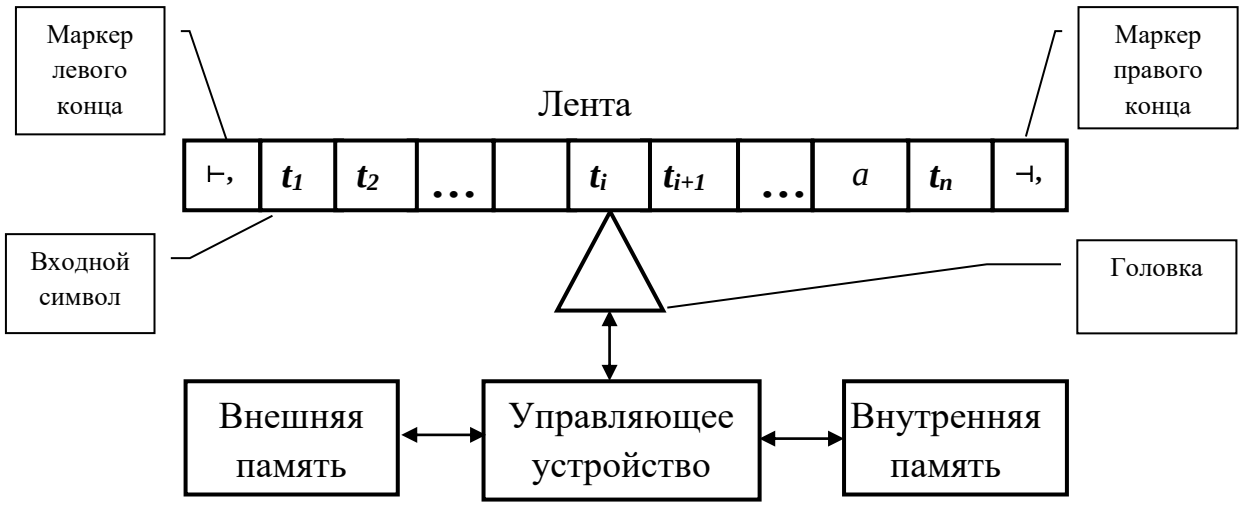


Рисунок II.2.1.1. Структура конечного автомата.

Конечный автомат может быть недетерминированным или детерминированным, но его головка должна быть односторонней

и может сдвигаться только вправо. Их формальные определения дается так:

Определение II.2.1.1. *Недетерминированный конечный автомат*(НКА) определяется семеркой $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{I}, \boldsymbol{F}, \vdash, \dashv, \Delta>$, где

$\boldsymbol{Q}$ – конечное множество состояний управляющего устройства;

$\boldsymbol{T}$ – конечное множество входных символов, $\boldsymbol{Q} \cap \boldsymbol{T} = \varnothing$;

$\boldsymbol{I}$ – множество начальных состояний управляющего устройства, $\boldsymbol{I} \subseteq \boldsymbol{Q}$;

$\boldsymbol{F}$ – множество конечных состояний управляющего устройства, указывающих, что входная цепочка распознается, $\boldsymbol{F} \subseteq \boldsymbol{Q}$;

$\vdash, \dashv$ – маркер начало, маркер конца ленты, $\vdash, \dashv \notin \boldsymbol{T}$;

Δ – множество отношений переходов $\Delta \subseteq \boldsymbol{Q} \times \boldsymbol{T}^* \times \vartheta(\boldsymbol{Q})$, $\vartheta(\boldsymbol{Q})$ – множество всех подмножеств множества $\boldsymbol{Q}$.

Детерминированный конечный автомат (ДКА) является частным случаем НКА.

Определение II.2.1.2. Конечный автомат $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{I}, \boldsymbol{F}, \vdash, \dashv, \Delta>$ называется *детерминированным*, если:

(1) множество начальных состояний $\boldsymbol{I}$ содержит ровно один элемент;

(2) для каждого перехода $<q, \tau, p> \in \Delta$ выполняется $|\tau| = 1$;

(3) для любого состояния $q \in \boldsymbol{Q}$ и для любого символа $t \in \boldsymbol{T}$ существует не более одного состояния $p \in \boldsymbol{Q}$ со свойством $<q, t, p> \in \Delta$;

(4) остальные символы остаются такими же, как и в НКА.

Замечания II.2.1.2:

1. Иногда вместо множества отношения переходов Δ, принимающего логическое значение ‘истина’ или ‘ложь’, используется функция переходов δ, принимающая значение в виде символа из множества $\boldsymbol{Q}$, где δ: $\boldsymbol{Q} \times \boldsymbol{T}^* \rightarrow \vartheta(\boldsymbol{Q})$ – в случае НКА и δ: $\boldsymbol{Q} \times \boldsymbol{T}^* \rightarrow \boldsymbol{Q}$ – в случае ДКА. От функции δ можно легко перейти к отношению Δ, положив

$$\Delta = \{<q, \tau, \delta(q, \tau)>: \ q \in \boldsymbol{Q}, \tau \in \boldsymbol{T}^*\}$$

2. В дальнейшем мы без особого упоминания, будем использовать как отношения переходов, так и функции переходов в зависимости от контекста. При этом для любых $q \in \boldsymbol{Q}$, $p \in \boldsymbol{Q}$ и $\tau \in \boldsymbol{T}^*$ можно писать:

1)для отношения переходов: $<q, \tau, \{p\}>$ – в случае НКА,$<q, \tau, p>$ – в случае ДКА;

2)для функции переходов: $\delta(q, \tau)=\{p\}$ – в случае НКА, $\delta(q, \tau) = p$ – в случае ДКА.

3. Если мы намереваемся вместо отношения переходов использовать функцию переходов, то в формальном определении КА следует символ Δ заменить на δ, а остальные символы оставить без изменения со своими прежними значениями, т.е. получим

$$\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{I}, \boldsymbol{F}, \vdash, \dashv, \delta>.$$

Переходы КА можно изображать в виде *диаграммы*, в которой каждое состояние обозначается кружком, а переход – стрелкой. Стрелка из состояния $q \in \boldsymbol{Q}$ в состояние $p \in \boldsymbol{Q}$ помеченная цепочкой $\tau \in \boldsymbol{T}^*$, показывает, что $<q, \tau, p>$ (или $\delta(q, \tau) = p$) является переходом данного НКА. Каждое начальное состояние распознается по ведущей в него короткой стрелке. Каждое заключительное состояние отмечается на диаграмме двойным кружком.

Примеры II.2.1.1:

1. Для КА $\boldsymbol{M}_1$ с одним переходом и параметрами: $\boldsymbol{Q}=\{q, p\}$; $\boldsymbol{T}^*=\{\tau\}$, $\boldsymbol{I} = \{q\}$, $\boldsymbol{F} = \{p\}$, $\delta(q, \tau) = p$ диаграмма показана на рисунке II.2.1.2.

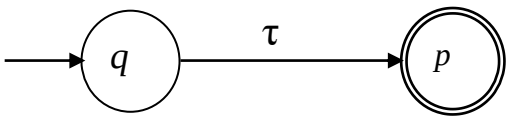


Рисунок II.2.1.2. Диаграмма КА $\boldsymbol{M}_1$ с одним переходом.

2. Пусть КА $\boldsymbol{M}_2$ имеет следующие параметры: $\boldsymbol{Q} = \{1,2\}$, $\mathbf{T} = \{a,b\}$, $\boldsymbol{I} =\{1\}$, $\boldsymbol{F} =\{2\}$, $\Delta = \{<1, aaa, 1>, <1, ab, 2>, <1, b, 2>, <2, \varepsilon, 1>\}$. Как видно из записи На рисунке II.2.1.3 показана диаграмма переходов

НКА $\boldsymbol{M}_2$, в которой в качестве меток ребер использованы регулярные выражения *aaa*, *ab*, *b*, ε. Такое представление облегчает построение диаграммы, делая её компактной и наглядной.

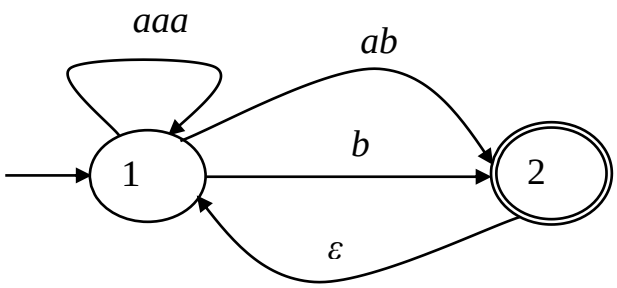


Рисунок II.2.1.3. Диаграмма КА $\boldsymbol{M}_2$ с регулярным выражение

3. КА $\boldsymbol{M}_3$ для распознавания идентификаторов, которые состоят только из букв и цифр и начинаются с буквы, будет иметь следующие параметры $\boldsymbol{Q}$ = {1, 2}, $\boldsymbol{T}$ = {*b*, *d*}, $\boldsymbol{I}$ = {1}, $\boldsymbol{F}$ = {2}, δ(*1*,*b*) =*2*, δ(*2*, *b*)=*2*, δ(*2*, *d*)=*2*, где *b* – буква, *d* – цифра. Диаграмма КА $\boldsymbol{M}_3$ представлена на рисунке *II*.2.1.4.

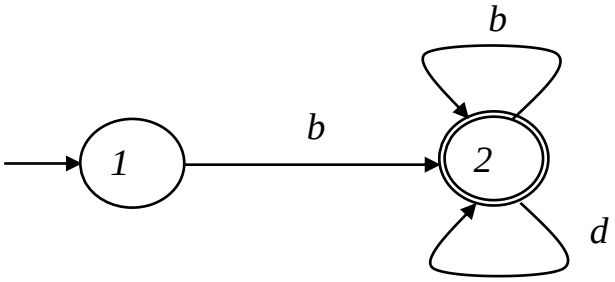


Рисунок II.2.1.4. Диаграмма КА $\boldsymbol{M}_3$ для идентификатора.

Замечание II.2.1.3. Если в диаграмме имеются несколько переходов с общим началом и концом, то эти переходы называются *параллельными*. Обычно параллельные переходы на диаграмме изображаются одной стрелкой. При этом метки переходов разделяются запятыми. На рисунке *II*.2.1.5 представлена диаграмма КА $\boldsymbol{M}_4$ с параллельными переходами для цепочек *ab*, *b*.

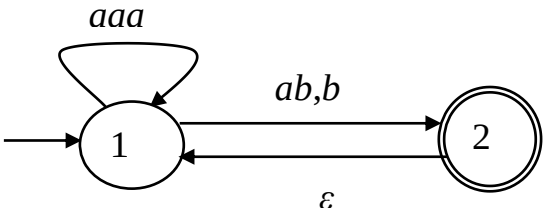


Рисунок II.2,1.5. Диаграмма КА $\boldsymbol{M}_4$. с параллельными переходами

Переходы КА можно представить в виде функции с помощью таблицы или команд.

Соглашение II.2.1.1. Среди всех состояний КА выделяются q_s начальное состояние и q_f финальное состояние, здесь *s* и *f* понимаются не как числовые переменные, а как мнемонические знаки начала (*start*) и конца (*final*).

Примеры II.2.1.3. В таблице *II*.2.1.1 показана функция переходов δ КА $\boldsymbol{M}_5$, определенная множествами $\boldsymbol{Q} = \{q_s, q_1, q_2, q_3\}$ и $\boldsymbol{T} = \{t_1, t_2, t_3\}$.

Таблица *II*.2.1.1. Значения функции переходов δ КА $\boldsymbol{M}_5$.

δ		Вход		
		t_1	t_2	t_3
Состояние	q_s	q_2	q_2	q_2
	q_1	q_3	q_s	q_s
	q_2	q_2	q_2	q_2
	q_3	q_3	q_2	q_s

Функция переходов из таблицы *II*.2.1.1 можно представить в виде команд следующим образом:

$$\delta(q_s, t_1) = q_2,\ \delta(q_s, t_2) = q_2,\ \delta(q_s, t_3) = q_2,$$
$$\delta(q_1, t_1) = q_3,\ \delta(q_1, t_2) = q_s,\ \delta(q_1, t_3) = q_s,$$
$$\delta(q_2, t_1) = q_2,\ \delta(q_2, t_2) = q_2,\ \delta(q_2, t_3) = q_2,$$
$$\delta(q_3, t_1) = q_3,\ \delta(q_3, t_2) = q_2,\ \delta(q_3, t_3) = q_s.$$

Пусть задан КА $\boldsymbol{M}$ с начальным состоянием $q_s \in \boldsymbol{Q}$, текущим состоянием $q \in \boldsymbol{Q}$, заключительным состоянием $q_f \in \boldsymbol{Q}$ и неиспользованной текущей входной цепочкой $\tau \in \boldsymbol{T}^*$. Тогда можно дать следующее определение.

Определения II.2.1.3:

1. Если головка обозревает самый левый символ входной цепочки τ, то пара $(q_s,\tau)\in \boldsymbol{Q}\times \boldsymbol{T}^*$ называется *начальной конфигурацией* КА;

2. Если головка обозревает текущий символ входной цепочки τ, то пара $(q,\tau)\in \boldsymbol{Q}\times \boldsymbol{T}^*$ называется *текущей конфигурацией* КА;

3. Если входная цепочка τ прочитана полностью, то пара $(q_f, \varepsilon)\in \boldsymbol{Q}\times \boldsymbol{T}^*$ называется *заключительной конфигурацией* КА;

Замечание II.2.1.4. Содержательно конфигурация представляет собой "мгновенное описание" КА. Если представить, что исходная цепочка, принадлежность которой рассматриваемому языку надо проверить, дано в ленте, то в конфигурации (q,τ) цепочка τ есть та часть исходной цепочки, которая пока осталась в ленте.

Такт КА определяется состоянием управляющего устройства и обозреваемым в этот момент входным символом. Сам такт состоит в изменении состояния управляющего устройства и сдвига головки на одну ячейку вправо.

Такт КА $\boldsymbol{M}$ задается бинарным отношением $\models_{\boldsymbol{M}}$, определенным над его конфигурациями из множества $\boldsymbol{Q}\times \boldsymbol{T}^*$. При этом, если известен автомат, то букву $\boldsymbol{M}$ при отношении $\models_{\boldsymbol{M}}$ можно опустить.

Пусть $t\in \boldsymbol{T}$ – еще не прочитанный самый левый символ входной цепочки и для $q\in \boldsymbol{Q}$, $p\in \boldsymbol{Q}$ выполняется $<q, t, p>\in \Delta$, то для цепочек $\tau\in \boldsymbol{T}^*$ выполняется отношение $(q, t\tau) \models (p, \tau)$, задающее такт автомата, которое означает, что автомат находясь в состоянии q и головка обозревает во входной ленте символ t, то КА $\boldsymbol{M}$ переходит в состояние p и головка сдвигается на ячейку вправо. При этом если $\tau = \varepsilon$, то считается входная цепочка *прочитан полностью*.

Примеры II.2.1.4. Пусть $\tau = abba$. Тогда в диаграмме КА $\boldsymbol{M_2}$ на рисунке *II*.2.1.3 имеется такт в виде отношения $(1, abba) \models (2, ba)$.

Определение II.2.1.4. $\models^k$ является k-той степенью отношения $\models$, если существует цепочка из $k+1$ конфигураций

$$(q_0,\tau_0), (q_1,\tau_1), (q_2,\tau_2),\ldots, (q_{k-1},\tau_{k-1}), (q_k,\tau_k)$$

такая, что для любого i $(1\le i\le k)$ выполняется отношение

$(q_{i-1},\tau_{i-1}) \models (q_i,\tau_i)$, где $q_0=q_s$, $\tau_0=\tau$, $q_k= q_f$, $\tau_k=\varepsilon$.

Если для любого $i\geq 1$ или $i\geq 0$ выполняется $(q_0,\tau) \models^i (q_i,\varepsilon)$, то можно записать $(q_0,\tau) \models^+ (q_i,\varepsilon)$ или $(q_0,\tau) \models^* (q_i,\varepsilon)$ соответственно. Здесь через $\models^+$ обозначается транзитивное замыкание отношения $\models$, а через $\models^*$ - рефлексивное и транзитовное замыкание отношения $\models$.

Определение II.2.1.5. Автомат ***M*** распознает входную цепочку τ, если выполняется отношение $(q_s, \tau) \models^* (q_f, \varepsilon)$.

Примеры II.2.1.5. Пусть τ = *aaaab*. Тогда в КА $\boldsymbol{M}_2$ на рисунке *II*.2.1.3 выполняются следующие отношения (1, *aaaab*) $\models$ (1, *ab*) и (1, *ab*) $\models$ (2, ε).

Определение II.2.1.6. Если язык ***L*** состоит только из распознанных автоматом ***M*** входных цепочек, то этот язык распознается автоматом ***M*** и он обозначается через ***L***(***M***), т.е.

$$\boldsymbol{L}(\boldsymbol{M}) \rightleftharpoons \{\tau:\ \tau\in \boldsymbol{T}^*\ \&\ (q_s, \tau) \models^* (q_f, \varepsilon)\}.$$

Лемма II.2.1.1. Если верно $(q_1,\ x) \models^* (q_2,\ \varepsilon)$ и $(q_2,\ y) \models^* (q_3,\ \varepsilon)$, то верно $(q_1,\ xy) \models^* (q_3,\ \varepsilon)$.

Доказательство. Для этого нужно провести индукцию по количеству тактов в программе работы КА, ведущей из конфигурации $(q_1,\ x)$ в конфигурацию $(q_2,\ \varepsilon)$.

Примеры II.2.1.6. Пусть для $\boldsymbol{M}_6$=<$\{q_s,q_1,q_f\}$,$\{0,1\}$,q_s,$\{q_f\}$,$\vdash$,$\dashv$,δ> имеются следующие отношения переходов:

<q_s,0,$\{q_1\}$>,<q_s,1,$\{q_s\}$>,<q_1,0,$\{q_f\}$>,<q_1,1,$\{q_s\}$>,<q_f,0,$\{q_f\}$>,<q_f,1,$\{q_f\}$>

КА $\boldsymbol{M}_6$ распознает всех цепочек из нулей и единиц, в составе которых имеются подряд два нуля. Здесь состояния можно интерпретировать следующим образом:

q_s – начальное состояние показывает, что «подряд стоящие два нуля еще не встретились и начальный символ является нулем»;

q_1 – состояние показывает, что «подряд стоящие два нуля еще не встретились и начальный символ является нулем»;

q_f – финальное состояние показывает, что «подряд стоящие два нуля уже появились».

Можно заметить, что КА $\boldsymbol{M}_6$ попав в состояние q_f остается в этом состоянии.

Для входной цепочки 01001 единственно возможная цепочка конфигураций, начиная с конфигурации (q_0, 01001) будет так

$(q_s, 01001) \models (q_1, 1001) \models (q_s, 001) \models (q_1, 01) \models (q_f, 1) \models (q_f, \varepsilon)$.

Итак, входная цепочка 01001 принадлежит языку, распознаваемому КА $\boldsymbol{M}_6$, т.е. $01001 \in \boldsymbol{L}(\boldsymbol{M}_6)$.

Диаграмма этого автомата показана на рисунке *II*.2.1.5.

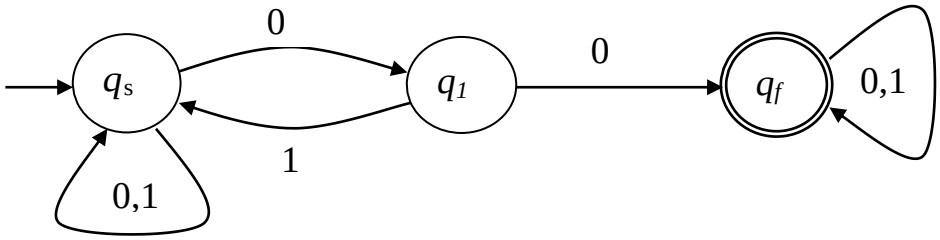


Рисунок II.2.1.5. Диаграмма КА $\boldsymbol{M}_6$.

Определения II.2.1.7:

1. *Путь* (path) КА - это кортеж $<q_0, r_1, q_1, r_2, \ldots, q_n>$, где $n \geq 0$ и $r_i = <q_{i-1}, \tau_i, q_i> \in \Delta$ для каждого i, $1 \leq i \leq n$. При этом q_0 - *начало пути*, q_n - *конец пути*, $\tau_1 \ldots \tau_n$ - *метка пути*, n - *длина пути*.

2. Путь называется *успешным* если его начало принадлежит $\boldsymbol{I}$, а конец принадлежит $\boldsymbol{F}$.

Замечание II.2.1.5. Для любого состояния $q \in \boldsymbol{Q}$ существует путь $<q>$. Его метка ε, начало и конец совпадают.

Примеры II.2.1.7. Рассмотрим КА $\boldsymbol{M_2}$ на рисунке *II*.2.1.3 Пусть $\tau = baaab$. Тогда путь <1,<1,*b*,2>,2,<2,ε,1>,1,<1,*aaa*,1>,1,<1,*b*,2>,2> является успешным. Его меткой является *baaab*, а длина равна 4, т.е.:

$q_0=1, q_1=2, q_2=1, q_3=1, q_4=2$;

$r_1=<1,b,2>, r_2=<2,\varepsilon,1>, r_3=<1,aaa,1>, r_4=<1,b,2>$;

$\tau_1=b, \tau_2=\varepsilon, \tau_3=aaa, \tau_4=b$.

Используя понятие «путь» можно дать альтернативные определения уже введенным выше понятиям распознаваемым цепочки и языка.

Определения II.2.1.8:

1. Цепочка $\tau \in \boldsymbol{T}^*$ *распознается* (*is recognized*) КА $\boldsymbol{M}$, если она является меткой некоторого успешного пути.

2. КА $\boldsymbol{M}$ *распознает* (*recognizes*) язык $\boldsymbol{L}(\boldsymbol{M})$, если он состоит только из меток всех успешных путей.

Замечание II.2.1.6. Если $\boldsymbol{I} \cap \boldsymbol{F} \neq \varnothing$, то язык, распознаваемый КА $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \vdash, \dashv, \boldsymbol{I}, \boldsymbol{F}, \Delta>$ содержит пустую цепочку ε.

Примеры II.2.1.8. Если КА $\boldsymbol{M_7} = <\boldsymbol{Q}, \boldsymbol{T}, \vdash, \dashv, \boldsymbol{I}, \boldsymbol{F}, \Delta>$ задан как $\boldsymbol{Q} = \{q_1, q_2\}$, $\boldsymbol{T} = \{a, b\}$, $\boldsymbol{I} = \{q_1\}$, $\boldsymbol{F} = \{q_1, q_2\}$, $\Delta = \{<q_1, a, q_2>, <q_2, b, q_1>\}$, то он является детерминированным и распознает следующий язык:

$$\boldsymbol{L}(\boldsymbol{M_7}) = \{(ab)^n : \ n \geq 0\} \cup \{(ab)^n a : \ n \geq 0\}.$$

Диаграмма этого автомата показана на рисунке *II*.2.1.6.

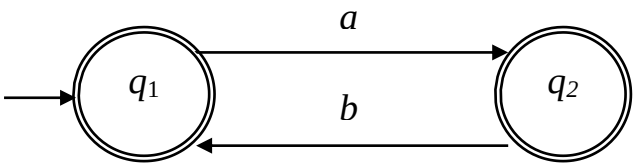


Рисунок II.2.1.6. Диаграмма КА $\boldsymbol{M_7}$.

Определение II.2.1.9. ДКА $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \vdash, \dashv, \boldsymbol{I}, \boldsymbol{F}, \Delta>$, называется *полным*, если для каждого состояния $q \in \boldsymbol{Q}$ и для каждого символа $t \in \boldsymbol{T}$ найдется такое состояние $p \in \boldsymbol{Q}$, что $<q, t, p> \in \Delta$, т.е. $\delta(q, t) = p$.

Примеры II.2.1.9. Диаграмма полного автомата $\boldsymbol{M_8}$ со следующими параметрами $\Delta = \{<1,a,2>, <1,b,3>, <2,a,3>, <2,b,1>, <3,a,3>, <3,b,3>\}$, $\boldsymbol{Q} = \{1,2,3\}$, $\boldsymbol{T} = \{a,b\}$, $q_s = \{1\}$, $\boldsymbol{F} = \{1,2\}$ показана на рисунке *II*.2.1.7.

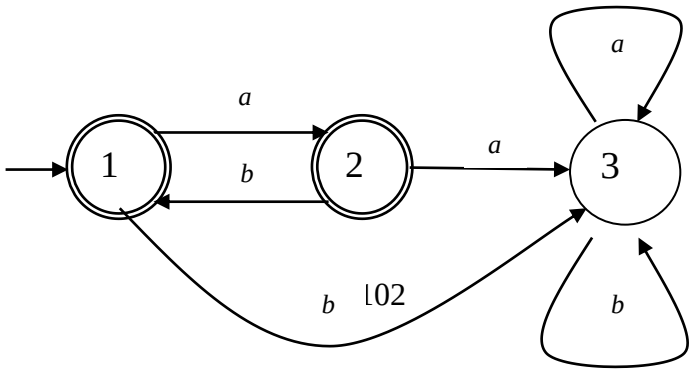


Рисунок II.2.1.7. Диаграмма КА $\boldsymbol{M_8}$

Задания II.2.1:

1. Найти КА, распознающий язык $\{\alpha\beta\colon\ \alpha\in\{a,b\}^*, \beta\in\{a,b\}^*\}$.
2. Найти КА, распознающий язык $\{a,b\}^* \setminus (\{a^n\colon\ n\geq 0\}\cup\{\ b^n\colon\ n\geq 0\})$.
3. Найти КА, распознающий язык $\{a\xi b\colon \xi\in\{a,b\}^*\cup\{b\xi a\colon\ \xi\in\{a,b\}^*\}$.
4. Найти КА, распознающий язык $\{\tau\in\{a,b\}^*\colon\ |\tau|_a\geq 3\}$.
5. Найти КА, распознающий язык $\{a^m b^n a^m b^n\colon\ m,n\geq 1\}$.
6. Перечислить все конфигурации $(q,\ \tau)$, удовлетворяющие условию $(1, abaacdcc) \models^* (q,\ \tau)$, в КА $\boldsymbol{M_9}$ на рисунке *II*.2.1.8.

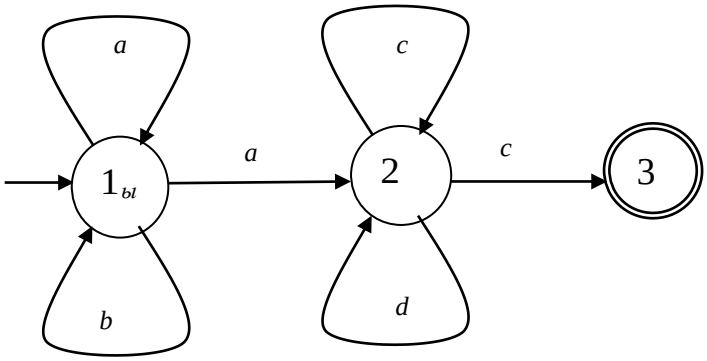


Рисунок II.2.1.8. Диаграмма КА $\boldsymbol{M_9}$.

7. Определить такт автомата, если он задан в следующем виде

$$\boldsymbol{M} = <\{\ q_0, q_1, q_2, q_f\}, \{a, b, c\}, \delta, q_0, \{q_f\}>,$$

где $\delta(q_0,a)=\{\ q_1, q_2\}$, $\delta(\ q_1,a)=\{q_1\}$, $\delta(q_1,b)=\{q_f\}$, $\delta(q_2,c)=\{q_f\}$,

$$\boldsymbol{L}(\boldsymbol{M}) = \{ac\}\cup\{a^n b\colon n\geq 1\}.$$

8. Найти полный детерминированный конечный автомат для языка $(a\vee b)^*(aab\vee abaa\vee abb)(a\vee b)^*$.
9. Найти полный детерминированный конечный автомат для языка $(b\vee c)((ab)^*c\vee(ba)^*)^*$.
10. Найти полный детерминированный конечный автомат для языка $(b\vee c)^*((a\vee b)^*c(b\vee a)^*)^*$.

Вопросы II.2.1:

1. Существуют ли КА, состояния q_1, q_2 и цепочки α, β, δ, такие что выполняются отношения $(q_1, \alpha\beta) \models^* (q_2, \beta)$ и $\neg (q_1, \alpha\delta) \models^* (q_2, \delta)$?

2. Как связаны $|\boldsymbol{Q}|$, $|\boldsymbol{T}|$, $|\Delta|$,$|\tau|$и число достижимых из (q, τ) конфигураций, в смысле $\models^*$?

3. Каким автоматом можно распознать язык, порожденный регулярным выражением $(abab) \vee (aba)^*$?

4. Является ли детерминированным КА $\boldsymbol{M_{10}}$ на рисунке II.2.1.9?

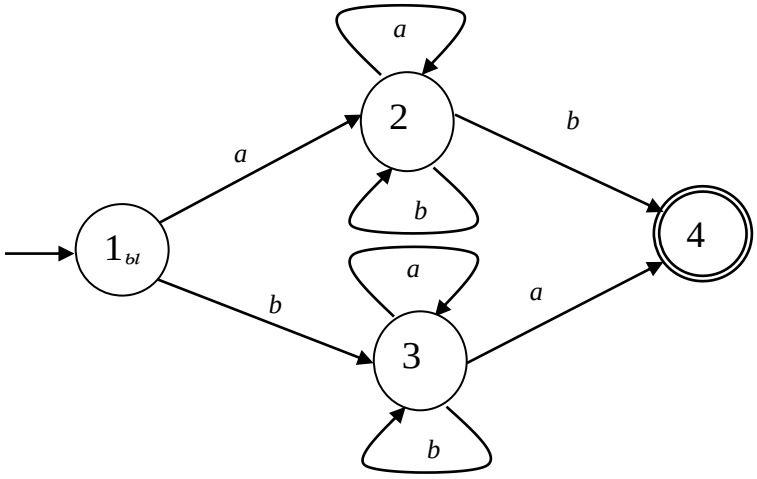


Рисунок II.2.1.9. Диаграмма КА $\boldsymbol{M_{10}}$.

5. Является ли полным детерминированный конечный автомат $\boldsymbol{M_{11}}$ с алфавитом $\boldsymbol{T} = \{a, b, c\}$ на рисунке II.2.1.10?

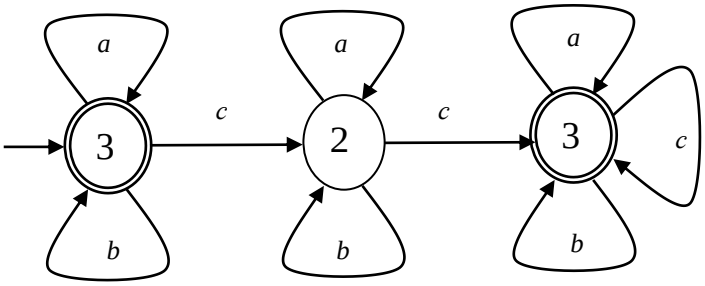


Рисунок II.2.1.10. Диаграмма КА $\boldsymbol{M_{11}}$.

6. Является ли полным детерминированный конечный автомат $\boldsymbol{M_{12}}$ с алфавитом $\boldsymbol{T} = \{a, b\}$ на рисунке II.2.1.11?

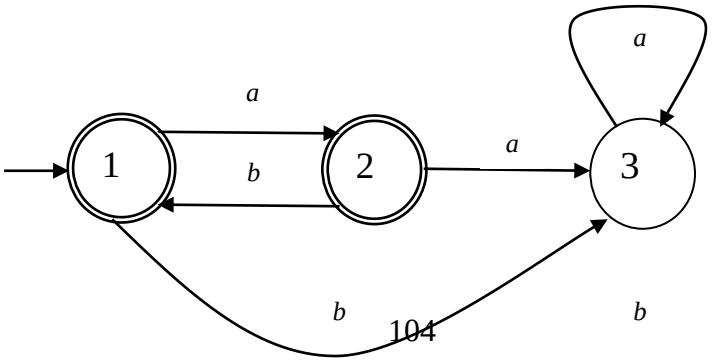


Рисунок II.2.1.11 Диаграмма КА $\boldsymbol{M_{12}}$.

II.2.2. Эквивалентность недетерминированых и детерминированных конечных автоматов

В этом параграфе будут обсуждены вопросы эквивалентности недетерминированного конечного автомата и детерминированного конечного автомата, распознающих один и тот же язык, а также будут предложены примеры, даны задания, сформулированы вопросы [1-3,5,10,13,19,23,25,26,32].

Один из наиболее важных научных результатов теории конечных автоматов состоит в том, что класс языков, распознаваемых недетерминированными конечными автоматами, совпадает с классом языков, определяемых детерминированными конечными автоматами. Это доказывается следующей теоремой.

Теорема II.2.2.1. Для любого НКА M, распознающего язык L, можно всегда построить ДКА M', распознающий этот же язык L.

Доказательство. Пусть задан НКА $M = <Q, T, \delta, F, \vdash, \dashv, \delta>$, распознающий язык $L=L(M)$, тогда ДКА $M' = <Q', T', q'_0, F', \vdash, \dashv, \delta'>$, распознающий этот же язык, т.е. $L=L(M')$, строится так:

(1) $Q' = \mathcal{P}(Q)$, т. е. состояниями ДКА M' являются множества состояний НКА M, где $\mathcal{P}(Q)$ – множество всех подмножеств множества Q;

(2) $q_0' = \{q_0\}$;

(3) множество F' состоит из всех таких подмножеств P множества Q, которые не пересекаются с множеством F, т.е.

$$F' = \{P: \forall P(P \subseteq \mathcal{P}(Q) \Rightarrow P \cap F = \varnothing)\};$$

(4) $T' = T$;

(5) $\delta(P,t)=P'$ для всех $P \subseteq Q$, где $P' = \{p: \delta(q, t) = p$, для некоторых $q \in P\}$ состояние p содержится в множестве $\delta(q, t)$, т.е. $\exists q(q \in P \& p \in \delta(q, t))$.

Состояние автомата M' обозначается через $[q_1, q_2, ..., q_i] \in Q'$, где $q_1, q_2, ..., q_i \in Q$; $q_0' = [q_0]$. Отсюда $\delta([q_1, q_2, ..., q_i], t) = [p_1, ..., p_j]$ тогда и только тогда, когда

$$(6)\delta([q_1, q_2, ..., q_k], t) = \bigcup_{i=1}^{k} \delta(q_i, t) = \{p_1, ..., p_j\}.$$

Сформулируем следующее утверждения: $(\boldsymbol{P},\tau) \models_{\boldsymbol{M}}^{i} (\boldsymbol{P}', \varepsilon)$ тогда и только тогда, когда $\boldsymbol{P}' = \{p: (q, \tau) \models_{\boldsymbol{M}}^{i} (p, \varepsilon)$ для некоторого $q \in \boldsymbol{P}\}$. Его Доказательство производится индукцией по i.

Базис индукции. Для i=0 очевидно, что $(q,\tau) \models_{\boldsymbol{M}}^{0} (p,\varepsilon)$ тогда и только тогда, когда $\tau = \varepsilon$ и $p \in \boldsymbol{P}'$.

Шаг индукции. Предположим, что $(\boldsymbol{P}, \tau) \models_{\boldsymbol{M}}^{i-1} (\boldsymbol{P}', \varepsilon)$ истинно для i, и возьмем $\tau = t\zeta$, где $|\zeta| = i$. Тогда $(\boldsymbol{P},\tau) \models_{\boldsymbol{M}}^{i} (\boldsymbol{P}', \varepsilon)$ равносильно тому, что $(q, t\zeta) \models_{\boldsymbol{M}}^{i} (p,\varepsilon)$ для некоторого $p \in \boldsymbol{Q}$.

Отсюда, в частности, следует, что $(\{q_0\},\tau) \models_{\boldsymbol{M}}^{i} (\boldsymbol{P}', \varepsilon)$ для некоторого $\boldsymbol{P}' \in \boldsymbol{F}'$ тогда и только тогда, когда $(q_0,\tau) \models_{\boldsymbol{M}}^{i} (p, \varepsilon)$ для некоторого $p \in \boldsymbol{F}$. Итак, $\boldsymbol{L}(\boldsymbol{M}') = \boldsymbol{L}(\boldsymbol{M})$.

Таким образом, функция перехода ДКА определяет только одно состояние, а функция перехода НКА – много состояний, т.е. если $q \in \boldsymbol{Q}$ текущее состояние, $p \in \boldsymbol{Q}$ – следующее состояние, $t \in \boldsymbol{T}$ – текущий входной символ, то $\delta(q, t)$=p – для ДКА, $\delta(q, t)$={p}– для НК и если $\delta(q, t) = \varnothing$, то функция $\delta(q, t)$ не определена.

Примеры II.2.2.1. Рассмотрим НКА $\boldsymbol{M_{13}} = <\boldsymbol{Q}, \boldsymbol{T}, \delta, q_0, \boldsymbol{F}, \vdash, \dashv, \delta>$ со следующими параметрами: $\boldsymbol{Q}$={1, 2, 3}, $\boldsymbol{T}$ = {a, b}, q_0 =1, $\boldsymbol{F}$ = {3}, $\delta(1, a) = 1$, $\delta(1, b) = 1$, $\delta(1, a) = 2$, $\delta(2, b) = 3$. Диаграмма НКА $\boldsymbol{M_{13}}$ показана на рисунке *II*.2.2.1.

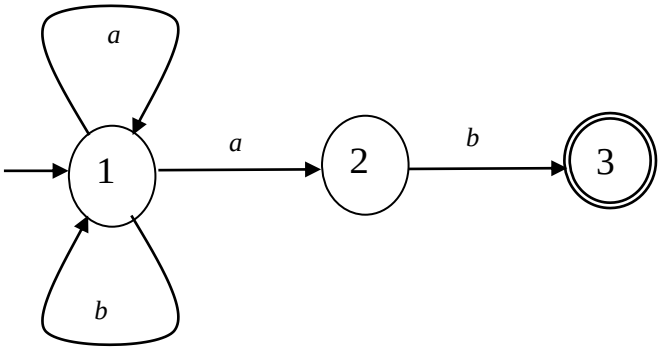


Рисунок II.2.2.1. Диаграмма НКА $\boldsymbol{M_{13}}$.

Для детерминизация этого НКА $\boldsymbol{M_{13}}$ применим теорему *II*.2.2.1. Если применить конструкцию из доказательства теоремы *II*.2.2.1 и затем удалить состояния, не достижимые из начального состояния, то получится полный ДКА $\boldsymbol{M_{14}}$ со следующими параметрами $\boldsymbol{Q}$ = {{1}, {1, 2}, {1, 3}}, $\boldsymbol{T}$ = {*a*,*b*}, q_0 ={1}, $\boldsymbol{F}$ ={1,3}, δ({1}, *a*)={1,2}, δ({1}, *b*)={1}, δ({1,2}, *a*) = {1,2}, δ({1,2}, *b*)={1,3}, δ({1,3}, *a*)={1,2}, δ({1,3}, *b*)={1}.

Диаграмма ДКА $\boldsymbol{M_{14}}$ показана на рисунке *II*.2.2.2.

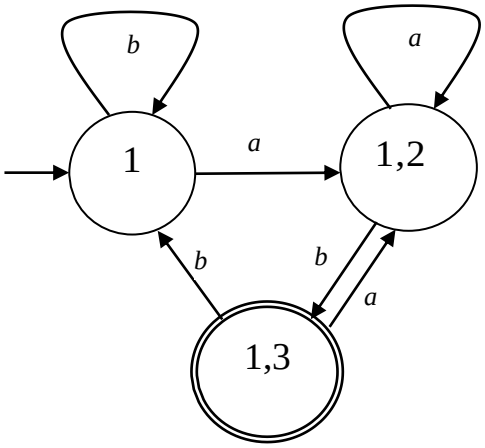


Рисунок II.2.2.2. Диаграмма ДКА $\boldsymbol{M_{14}}$.

Задания II.2.2:

1. Реализовать алгоритм, строящий ДКА из НКА таким образом, что бы входными данными была таблица состояний и переходов НКА, а на выходе таблица с состояниями и переходами ДКА.
2. Найти полный ДКА, эквивалентный следующему НКА $\boldsymbol{M_{15}}$.

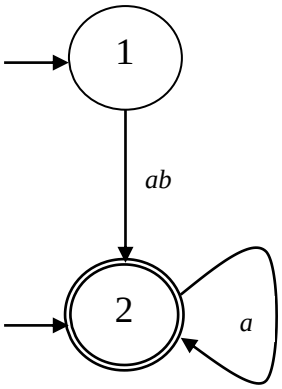


Рисунок II.2.2.3. Диаграмма НКА $\boldsymbol{M_{15}}$.

3. Построить детерминированный конечный автомат, распознающий язык $\{a\xi b\colon\ \xi\in\{a,b\}^*\cup\{b\xi a\colon\ \xi\in\{a,b\}^*\}$.

4. Найдите конечный автомат среди следующих:

$$\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{I}, \boldsymbol{F}, \vdash, \dashv, \Delta> \quad \text{и} \quad \boldsymbol{G}=<\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>.$$

5. Найдите функцию перехода для недетерминированного конечного автомата среди следующих:

$\delta(q,a)=p$;

$\delta(q,a)=\{p\}$;

$\delta(q,a)=\varnothing$;

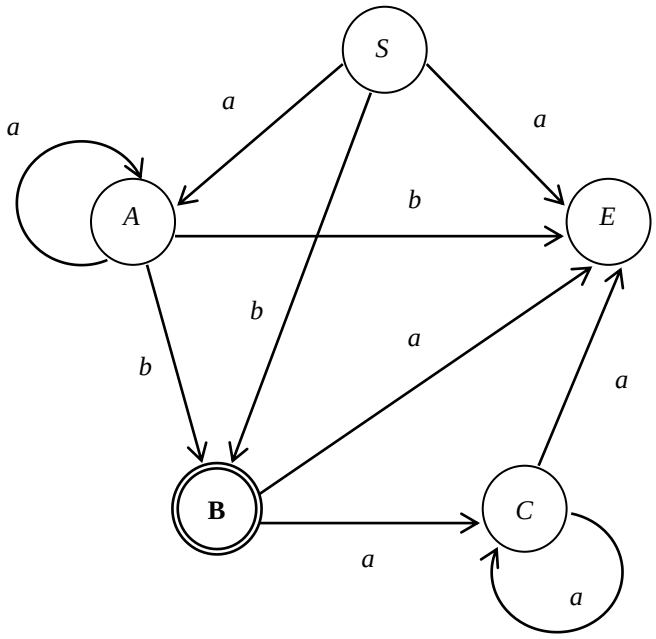
$\delta(q,a)=q$.

Вопросы II.2.2:

1. Что такое функция перехода конечного автомата?
2. Из чего состоит конфигурация конечного автомата?
3. Как определяется язык, распознаваемый автоматом?
4. Существуют ли конечный автомат $\boldsymbol{M}$ с такими состояниями q_1, q_2 и входными словами α, β, δ, что выполняются отношения:

$$(q_1, \alpha\,\beta) \models^* (q_2, \beta) \ \text{ и } \ \neg((q_1, \alpha\,\delta) \models^* (q_2, \delta)) \ ?$$

5. Какому (детерминированному или недетерминированному) конечному автомату относится следующая диаграмма?



II.3. Свойства регулярных языков

II.3.1. Эквивалентность регулярных выражений и праволинейных грамматик

В этом параграфе показывается, что класс регулярных языков, порождаемых регулярными выражениями, в точности совпадает с классом праволинейных языков, порождаемых праволинейными грамматиками, предложены примеры, даны задания, сформулированы вопросы [1-3,6,8,11-13,16-21,25,28-32].

Доказательство указанной выше эквивалентности производится построением праволинейной грамматики (ПЛГ), соответствующей заданному регулярному выражению (РВ), и наоборот, построением РВ, соответствующего заданной ПЛГ с помощью лемм II.3.1.1 и II.3.1.2.

Лемма II.3.1.1. Для любого РВ ρ, порождаемого регулярный язык $\boldsymbol{L}(\rho)$, можно построить ПЛГ $\boldsymbol{G}$, порождающей праволинейный язык $\boldsymbol{L(G)}$, такой, что $\boldsymbol{L(G)} \subseteq \boldsymbol{L}(\rho)$.

Доказательство. ПЛГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, порождающая язык $\boldsymbol{L(G)}$, строится индукцией по длине ρ – общему количеству символов алфавита $\boldsymbol{T}$, символов ϕ, ε, знаков операций ∨ - селекция (выбор), · - конкатенация (сцепление), * - итерация (повторение) и скобок в записи РВ ρ.

Базис индукции. Пусть длина РВ **ρ** равна 1. Тогда ρ должно быть одним из трех РВ (1) ϕ, (2) ε и (3) t для некоторого $t \in \boldsymbol{T}$, которые порождают регулярные языки $\varnothing$, $\{\varepsilon\}$ и $\{t\}$ соответственно.

Праволинейные грамматики, соответствующие РВ (1) ϕ, (2) ε и (3) t, будут иметь:

(1) Если $\boldsymbol{N} = \{S\}$, $\boldsymbol{P} = \varnothing$, то $\boldsymbol{L}(\boldsymbol{G}) = \varnothing$;

(2) Если $\boldsymbol{N} = \{S\}$, $\boldsymbol{P} = \{S \to \varepsilon\}$, то $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{\varepsilon\}$;

(3) Если $\boldsymbol{N} = \{S\}$, $\boldsymbol{P} = \{S \to t\}$, то $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{t\}$.

Индукционный шаг. Предположим, что для РВ ρ' и ρ'' длины $k \geq 1$, порождающие регулярные языки $\boldsymbol{L}(\rho')$ *и* $\boldsymbol{L}(\rho'')$, $\boldsymbol{L}(\rho') \cap \boldsymbol{L}(\rho'') = \varnothing$,

построены ПЛГ $\boldsymbol{G'}=<\boldsymbol{N'},\boldsymbol{T},\boldsymbol{P'},S'>$ и $\boldsymbol{G''}=<\boldsymbol{N''},\boldsymbol{T},\ \boldsymbol{P''},S''>$ такие, что $\boldsymbol{L(G')} \subseteq \boldsymbol{L}(\rho')$ и $\boldsymbol{L(G'')} \subseteq \boldsymbol{L}(\rho'')$.

Рассмотрим произвольное РВ ρ длины k+1. Это РВ может иметь один из трех следующих видов:

(1) $\rho = \rho' \vee \rho''$, порождает язык $\boldsymbol{L(\rho)} = \boldsymbol{L}(\rho') \cup \boldsymbol{L}(\rho'')$;

(2) $\rho = \rho' \cdot \rho''$, порождает язык $\boldsymbol{L(\rho)} = \boldsymbol{L}(\rho') \cdot \boldsymbol{L}(\rho'')$;

(3) $\rho = \rho'^*$, порождает язык $\boldsymbol{L(\rho)} = \boldsymbol{L}(\rho')^*$.

Тогда ПЛГ $\boldsymbol{G}=<\boldsymbol{N},\boldsymbol{T},\boldsymbol{P},S>$, порождающий язык $\boldsymbol{L(G)} \subseteq \boldsymbol{L}(\rho)$, строится следующим образом:

(***I***) Для РВ $\rho = \rho' \cup \rho''$:

$\boldsymbol{N} = \boldsymbol{N'} \cup \boldsymbol{N''} \cup S$, где S – новый начальный нетерминал;

$\boldsymbol{P} = \boldsymbol{P'} \cup \boldsymbol{P''} \cup \{S \rightarrow S' | S''\}$.

Язык $\boldsymbol{L(G)}$ будет включать все цепочки из языков $\boldsymbol{L(G')}$ и $\boldsymbol{L(G'')}$, что позволяет установить

$$\boldsymbol{L}(\boldsymbol{G})=\boldsymbol{L}(\boldsymbol{G'}) \cup \boldsymbol{L}(\boldsymbol{G''}) \subseteq \boldsymbol{L}(\rho') \cup \boldsymbol{L}(\rho'') = \boldsymbol{L}(\rho).$$

(***II***) Для РВ $\rho = \rho' \cdot \rho''$:

$\boldsymbol{N} = \boldsymbol{N'} \cup \boldsymbol{N''}$, $S = S'$, $\boldsymbol{P} = \boldsymbol{P'} \cup \boldsymbol{P''}$.

Язык $\boldsymbol{L(G)}$ будет включать все цепочки из $\boldsymbol{L(G')}$ и $\boldsymbol{L(G'')}$, что позволяет установить $\boldsymbol{L}(\boldsymbol{G})=\boldsymbol{L}(\boldsymbol{G'}) \cdot \boldsymbol{L}(\boldsymbol{G''}) \subseteq \boldsymbol{L}(\rho') \cdot \boldsymbol{L}(\rho'') = \boldsymbol{L}(\rho)$.

(***III***) Для РВ $\rho = \rho'^*$:

$\boldsymbol{N} = \boldsymbol{N'} \cup \boldsymbol{N''}$, $S = S'$, $\boldsymbol{P} = \boldsymbol{P'} \cup \{S \rightarrow \varepsilon | S\}$.

Язык $\boldsymbol{L(G)}$ будет включать все цепочки из $\boldsymbol{L(G')}$ и пустую цепочку ε, что позволяет установить $\boldsymbol{L}(\boldsymbol{G})=\boldsymbol{L}(\boldsymbol{G'})^* \subseteq \boldsymbol{L}(\rho')^* = \boldsymbol{L}(\rho)$.

Известно, что класс регулярных языков, содержащий множества $\varnothing$, $\{\varepsilon\}$ и $\{t\}$ для всех t в конечном алфавите $\boldsymbol{T}$ является замкнутым относительно операций объединения, конкатенации и итерации. По базису индукции регулярные множества $\varnothing$, $\{\varepsilon\}$ и $\{t\}$ являются праволинейными, следовательно, класс праволинейных языков замкнут относительно операций объединения, конкатенации и итерации.

Лемма II.3.1.2. Для любой ПЛГ $\boldsymbol{G}$, порождаемой праволинейный язык $\boldsymbol{L}(\boldsymbol{G})$, можно построить РВ ρ, порождающей регулярный язык $\boldsymbol{L}(\rho)$, такой, что $\boldsymbol{L}(\rho) \subseteq \boldsymbol{L}(\boldsymbol{G})$.

Доказательство. В качестве упражнения для читателя.

Из леммы II.3.1.1 и II.3.1.2 вытекает доказателство теоремы.

Теорема II.3.1. Класс языков, порождаемых праволинейными грамматиками совпадает с классом регулярных языков.

Примеры II.3.1.1. Пусть грамматика G определяется правилами $S \to 0A \mid 1S \mid \varepsilon$, $A \to 0B \mid 1A$, $S \to 0S \mid 1B$. Тогда соответствующая система уравнений получается из системы, если X_1, X_2, X_3 заменить соответственно на S, A, B. $L(G)$ - это множество цепочек, в которых число нулей делится на 3, и обозначается регулярным выражением.

Задания II.3.1. Построить праволинейная грамматики, эквивалентные регулярным выражениям:

1. $(a \vee b \vee ab)^*$.
2. $(a^*b)^* \vee (b^*a)^*$.
3. $(ba \vee a^*ab)^*$.
4. $(b^+a)^*b \vee a)b^*$.
5. $(b^*a)b \vee (ab^*)a$.
6. $a^*bc \vee b^*a$.
7. $(01^* \vee 1)^*11$

Вопросы II.3.1:

1. Для двух регулярных выражений, порождающих один и тот же язык, найдется ли праволинейная, которая будет эквивалентна обоим регулярным выражениям?

2. Будут ли равными между собой грамматики, эквивалентные регулярным выражениям $(0 \vee (1(0^*)))$ и $0 \vee 10^*$?

3. Как строится регулярное выражение, эквивалентное праволинейной грамматике $S \to E$, $S \to bE$, $S \to caE$, $E \to a$, $E \to bS$?

4. Равны ли праволинейные грамматики, эквивалентные регулярным выражениям $(a \vee b)^*(b \vee a)^*$ и $(a \vee b \vee ab)^*$?

5. Равны ли праволинейные грамматики, эквивалентные регулярным выражениям $(ab \vee b)^*$ и $(a \vee b)^*$?

II.3.2. Эквивалентность регулярных выражений и конечных автоматов

В этом параграфе показывается, что класс регулярных языков, порождаемых регулярными выражениями, в точности совпадает с классом языков, распознаваемых недетерминированными конечными автоматами, предложены примеры, даны задания, сформулированы вопросы [1-3,5-13,15-21,23,25-34].

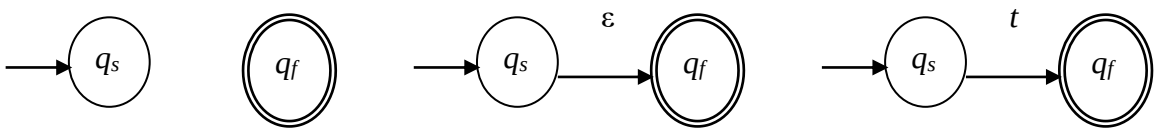
Это доказывается построением недетерминированного конечного автомата (НКА), соответствующего заданному регулярному выражению (РВ), и наоборот, построением РВ, соответствующего заданному НКА.

Лемма II.3.2.1. Для любого РВ ρ, порождаемого регулярный язык $\boldsymbol{L}(\rho)$, можно построить соответствующий НКА $\boldsymbol{M}$, распознающий язык $\boldsymbol{L}(\boldsymbol{M})$, такой, что $\boldsymbol{L}(\boldsymbol{M}) \subseteq \boldsymbol{L}(\rho)$.

Доказательство. НКА $\boldsymbol{M}$, распознающий язык $\boldsymbol{L}(\boldsymbol{M})$, строится индукцией по длине РВ ρ, т.е. по общему количеству символов алфавита $\boldsymbol{T}$, символов ϕ, ε, знаков операций ∨ - селекции, · - конкатенации, * - итерации и скобок в записи ρ.

Базис индукции. Пусть длина РВ ρ равна 1. Тогда ρ должно быть одним из трех РВ (1) ϕ, (2) ε и (3) t для некоторого $t \in \boldsymbol{T}$, порождающие регулярные языки ∅, $\{\varepsilon\}$ и $\{t\}$ соответственно.

НКА $\boldsymbol{M} = \langle \boldsymbol{Q}, \boldsymbol{T}, \vdash, \dashv, q_s, \boldsymbol{F}, \delta \rangle$, соответствующие РВ (1) ϕ, (2) ε и (3) t, будут иметь: $\boldsymbol{Q} = \{q_s, q_f\}$, q_s – начальное состояние, $q_f \in \boldsymbol{F}$ – заключительное состояние, $\boldsymbol{T} = \{t\}$. Верно отношение перехода $\langle q_s, t, q_f \rangle$, т.е. функция перехода $\delta(q_s, t) = q_f$, а в остальных случаях δ не определена. Следовательно $\boldsymbol{L}(\boldsymbol{M}) \rightleftharpoons \{t\}$. Диаграммы НКА, распознающие языки ∅, $\{\varepsilon\}$ и $\{t\}$ показаны на рисунке II.3.2.1.



(1) (2) (3)

Рисунок II.3.2.1. НКА, распознающие языки ∅, $\{\varepsilon\}$ и $\{t\}$.

Индукционный шаг. Предположим, что для каждого РВ ρ' и ρ'' длины $k \geq 1$, порождающие регулярные языки $\boldsymbol{L}(\rho')$ и $\boldsymbol{L}(\rho'')$, $\boldsymbol{L}(\rho') \cap \boldsymbol{L}(\rho'') = \emptyset$, построены соответствующие НКА $\boldsymbol{M}'$ и $\boldsymbol{M}''$. Рассмотрим произвольное РВ ρ длины $k+1$. Это РВ может иметь один из трех следующих видов:

(1) $\rho = \rho' \vee \rho''$, порождает язык $\boldsymbol{L}(\rho) = \boldsymbol{L}(\rho') \cup \boldsymbol{L}(\rho'')$;

(2) $\rho = \rho' \cdot \rho''$, порождает язык $\boldsymbol{L}(\rho) = \boldsymbol{L}(\rho') \cdot \boldsymbol{L}(\rho'')$;

(3) $\rho = \rho'^*$, порождает язык $\boldsymbol{L}(\rho) = \boldsymbol{L}(\rho')^*$.

Пусть $\boldsymbol{M}' = \langle \boldsymbol{Q}', \boldsymbol{T}, q'_s, \boldsymbol{F}', \vdash, \dashv, \delta' \rangle$ и $\boldsymbol{M}'' = \langle \boldsymbol{Q}'', \boldsymbol{T}, q''_s, \boldsymbol{F}'', \vdash, \dashv, \delta'' \rangle$ НКА, распознающие языки $\boldsymbol{L}(\boldsymbol{M}') = \boldsymbol{L}(\rho')$ и $\boldsymbol{L}(\boldsymbol{M}'') = \boldsymbol{L}(\rho'')$.

Не ограничивая общности, мы будем предполагать, что у них разные состояния, т.е. $\boldsymbol{Q}' \cap \boldsymbol{Q}'' = \emptyset$, так как в противном случае состояния можно было бы переименовать. Тогда недетерминированный конечный автомат $\boldsymbol{M} = \langle \boldsymbol{Q}, \boldsymbol{T}, \vdash, \dashv, \boldsymbol{I}, \boldsymbol{F}, \Delta \rangle$, распознающий язык $\boldsymbol{L}(\boldsymbol{M}) \subseteq \boldsymbol{L}(\rho)$ строится следующим образом:

($\boldsymbol{I}$) Для РВ $\rho = \rho' \vee \rho''$:

$\boldsymbol{Q} = \boldsymbol{Q}' \cup \boldsymbol{Q}'' \cup \{q_s, q_f\}$, где q_s – новое начальное состояние, q_f – новое заключительное состояние;

$$\boldsymbol{F} = \begin{cases} \boldsymbol{F}' \cup \boldsymbol{F}'', \text{ если } \varepsilon \notin \boldsymbol{L}(\rho') \cup \boldsymbol{L}(\rho''); \\ \boldsymbol{F} = \boldsymbol{F}' \cup \boldsymbol{F}'' \cup \{q_f\}, \text{ если } \varepsilon \in \boldsymbol{L}(\rho') \cup \boldsymbol{L}(\rho''); \end{cases}$$

$\delta(q_s, t) = \delta(q'_s, t) \cup \delta(q''_s, t)$ для всех $t \in \boldsymbol{T}$;

$$\delta(q, t) = \begin{cases} \delta'(q, t) \text{ для всех } q \in \boldsymbol{Q}' \text{ и } t \in \boldsymbol{T}, \\ \delta''(q, t) \text{ для всех } q \in \boldsymbol{Q}'' \text{ и } t \in \boldsymbol{T}. \end{cases}$$

Кроме того, в δ будут содержаться и четыре ε-переходов:

$$\delta(q_s, \varepsilon) = q'_s, \ \delta(q_s, \varepsilon) = q''_s, \ \delta(q'_f, \varepsilon) = q_f, \ \delta(q''_f, \varepsilon) = q_f.$$

Язык $\boldsymbol{L}(\boldsymbol{M})$ будет включать все цепочки из $\boldsymbol{L}(\boldsymbol{M}')$ и $\boldsymbol{L}(\boldsymbol{M}'')$. Сначала НКА $\boldsymbol{M}$ угадывает, какой из НКА $\boldsymbol{M}'$ и $\boldsymbol{M}''$ ему моделировать. Затем НКА $\boldsymbol{M}$ в силу своей недетерминированности фактически моделирует и $\boldsymbol{M}'$, и $\boldsymbol{M}''$. Можно показать индукцией по $k \geq 1$, что $(q_s, \tau) \models^k_{\boldsymbol{M}} (q, \varepsilon)$ тогда и только тогда, когда выполняется $q \in \boldsymbol{Q}'$ и $(q'_s, \tau) \models^k_{\boldsymbol{M}'} (q, \varepsilon)$ или $q \in \boldsymbol{Q}''$ и

$(q''_s,\ \tau) \models^k_{M''} (q,\ \varepsilon)$. Отсюда $\boldsymbol{L}(\boldsymbol{M})=\boldsymbol{L}(\boldsymbol{M}')\cup\boldsymbol{L}(\boldsymbol{M}'')$. Диаграмма НКА $\boldsymbol{M}$ показана на рисунке II.3.2.2.

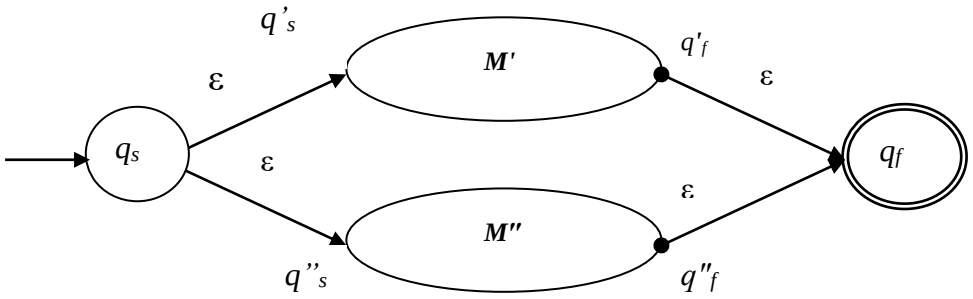


Рис. II.3.2.2. НКА $\boldsymbol{M}$, распознающий язык $\boldsymbol{L}(\boldsymbol{M})=\boldsymbol{L}(\boldsymbol{M}')\cup\boldsymbol{L}(\boldsymbol{M}'')$

Здесь каждая цепочка $\tau\in\boldsymbol{L}(\boldsymbol{M})$ переводит q_s в q_f, и после первого шага несущий её путь проходит через q'_s или q''_s. Так как состояния $\boldsymbol{M}'$ и $\boldsymbol{M}''$ не пересекаются, то в первом случае этот путь может попасть в q_f только по ε-переходу из q'_f и тогда $\tau\in\boldsymbol{L}(\boldsymbol{M}')$. Аналогично, во втором случае попасть в q_f только по ε-переходу из q''_f и $\tau\in\boldsymbol{L}(\boldsymbol{M}'')$.

(***II***) Для РВ $\rho = \rho'\cdot\rho''$:

$\boldsymbol{Q} = \boldsymbol{Q}'\cup\boldsymbol{Q}''$;

$q_s = q'_s\in\boldsymbol{Q}'$ – начальное состояние;

$$\boldsymbol{F} = \begin{cases} \boldsymbol{F}'', \text{ если } q''\notin\boldsymbol{F}''; \\ \boldsymbol{F}'\cup\boldsymbol{F}'', \text{ если } q''\in\boldsymbol{F}''; \end{cases}$$

функция перехода δ определяется равенствами:

(1) $\delta(q, t)= \delta'(q, t)$ для всех $q\in\boldsymbol{Q}'\setminus\boldsymbol{F}'$ и $t\in\boldsymbol{T}$;

(2) $\delta(q, t)= \delta'(q, t)\cup\{\delta(q, \varepsilon)=q''_s\}\cup\delta''(q''_s, t)$ для всех $q\in\boldsymbol{F}'$ и $t\in\boldsymbol{T}$;

(3) $\delta(q, t)= \delta''(q, t)$ для всех $q\in\boldsymbol{Q}''$ и $t\in\boldsymbol{T}$.

Таким образом, автомат $\boldsymbol{M}$ начинает работать, моделируя $\boldsymbol{M}'$. Когда автомат $\boldsymbol{M}$ достигает заключительного состояния автомата $\boldsymbol{M}'$ он через ε-переход $\delta(q'_f, \varepsilon) = q''_s$ перейдет в начальное состояние автомата $\boldsymbol{M}''$ и моделировать $\boldsymbol{M}''$.

Пусть даны две цепочки $\chi\in\boldsymbol{L}(\boldsymbol{M}')$ и $\xi\in\boldsymbol{L}(\boldsymbol{M}'')$. Тогда выполняется $(q'_s, \chi\xi) \models^+_м (q, \xi)$ для некоторого $q\in\boldsymbol{F}'$. Если $\chi=\varepsilon$, то $q=q'$. Если $\xi\neq\varepsilon$, то, применяя один раз равенство (2) и нуль или более раз равенство (3), получаем $(q, \xi) \models^+_м (p, \varepsilon)$ для некоторого $p\in\boldsymbol{F}''$. Если $\xi=\varepsilon$, то $q\in\boldsymbol{F}$, так

как $p \in \boldsymbol{F}''$. Отсюда $\chi\xi \in \boldsymbol{L}(\boldsymbol{M})$. Допустим, что $\tau \in \boldsymbol{L}(\boldsymbol{M})$. Тогда $(q', \tau) \models^*_{\text{м}} (q_f, \varepsilon)$. Рассмотрим отдельно два случая: $q_f \in \boldsymbol{F}''$ и $q_f \in \boldsymbol{F}'$. Пусть $q_f \in \boldsymbol{F}''$. Тогда $\tau = \chi t \xi$ для некоторого $t \in \boldsymbol{T}$, удовлетворяющего условиям $(q'_s, \chi t \xi) \models^* (q'_f, t\xi) \models (q''_s, \xi) \models^* (q''_f, \varepsilon)$. Следовательно, $\chi \in \boldsymbol{L}(\boldsymbol{M}')$ и $t\xi \in \boldsymbol{L}(\boldsymbol{M}'')$. Пусть $q_f \in \boldsymbol{F}'$, тогда $\varepsilon \in \boldsymbol{L}(\boldsymbol{M}'')$. Таким образом, $\tau \in \boldsymbol{L}(\boldsymbol{M}')$. Отсюда $\boldsymbol{L}(\boldsymbol{M}) = \boldsymbol{L}(\boldsymbol{M}') \cdot \boldsymbol{L}(\boldsymbol{M}'')$. Диаграмма НКА $\boldsymbol{M}$ представлена на рисунке II.3.2.3.

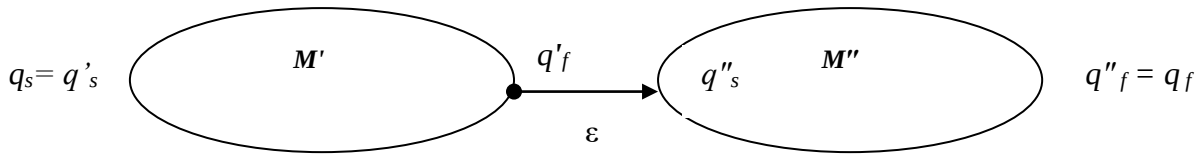


Рисунок II.3.2.3. НКА $\boldsymbol{M}$, распознающий язык $\boldsymbol{L}(\boldsymbol{M}) = \boldsymbol{L}(\boldsymbol{M}') \cdot \boldsymbol{L}(\boldsymbol{M}'')$.

Здесь каждая цепочка $\tau \in \boldsymbol{L}(\boldsymbol{M})$ переводит q_s в q_f, и после первого шага несущий её путь проходит через q'_s или q''_s. Так как состояния $\boldsymbol{M}'$ и $\boldsymbol{M}''$ не пересекаются, то в первом случае этот путь может попасть в q_f только по ε-переходу из q'_f и тогда $\tau \in \boldsymbol{L}(\boldsymbol{M}')$. Аналогично, во втором случае попасть в q_f только по ε-переходу из q''_f и $\tau \in \boldsymbol{L}(\boldsymbol{M}'')$.

(***III***) Для РВ $\rho = \rho'^*$:

$\boldsymbol{Q} = \boldsymbol{Q}' \cup \{q_s, q_f\}$, где q_s – начальное состояние, q_f – заключительное состояние, $\boldsymbol{F} = \boldsymbol{F}' \cup \{q_f\}$, функция перехода δ определяется равенствами:

(1)$\delta(q, t) = \delta'(q, t)$, если $q \in \boldsymbol{Q} \setminus \boldsymbol{F}'$ и $t \in \boldsymbol{T}$,

(2)$\delta(q, t) = \delta'(q, t) \cup \delta'(q'_s, t)$, если $q \in \boldsymbol{F}'$ и $t \in \boldsymbol{T}$,

(3)$\delta(q_s, t) = \delta'(q'_s, t)$ для $t \in \boldsymbol{T}$.

Итак, когда $\boldsymbol{M}$ попадает в заключительное состояние автомата $\boldsymbol{M}'$, он может недетерминированно выбрать либо продолжать моделирование $\boldsymbol{M}'$, либо начать заново моделирование $\boldsymbol{M}'$ с начального состояния. Доказательство того, что $\boldsymbol{L}(\boldsymbol{M}) = \boldsymbol{L}(\rho)^*$, аналогично доказательству части (***II***). Заметим, что $\varepsilon \in \boldsymbol{L}(\boldsymbol{M})$, так как $q' \in \boldsymbol{F}'$ – заключительное состояние. Функция перехода НКА $\boldsymbol{M}$ будет включать, помимо функции перехода автомата $\boldsymbol{M}'$, четыре новых ε-

переходов: $\delta(q_s, \varepsilon)=q'_s$, $\delta(q_s, \varepsilon)=q_f$, $\delta(q'_f, \varepsilon)=q'_s$, $\delta(q'_f, \varepsilon)=q_f$ Диаграмма этого автомата представлена на рисунке II.3.2.4.

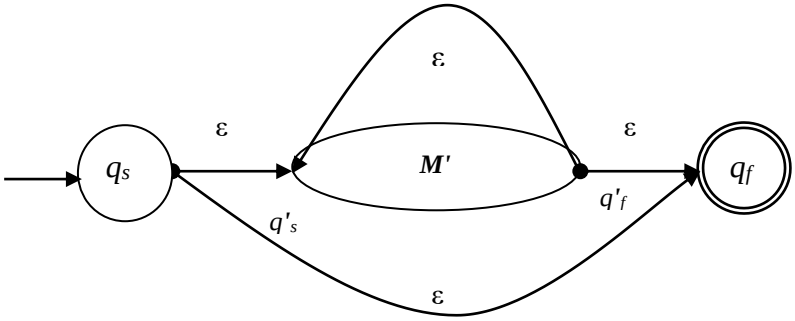


Рис. II.3.2.4. Диаграмма КА $\boldsymbol{M}$, распознающего язык $\boldsymbol{L}(\boldsymbol{M'})$*.

Здесь для непустой цепочки $\tau \neq \varepsilon$ по определению итерации $\tau \in \boldsymbol{L}(\rho')^*$ для $k \geq 1$ цепочку τ можно разбить на k подцепочек: $\tau = \tau_1\tau_2\ldots\ \tau_k$ и все $\tau_i \in \boldsymbol{L}(\boldsymbol{M'})$. Для каждого $i = 1,\ldots, k$ цепочка τ_i переводит состояние q'_s в состояние q'_f. Тогда для цепочки τ в диаграмме $\boldsymbol{M}$ имеется путь, представленный на рисунке II.3.2.5.

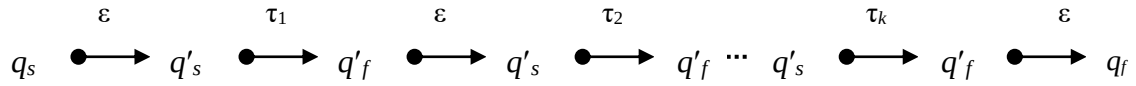


Рисунок II.3.2.5. Путь в диаграмме КА $\boldsymbol{M}$.

Следовательно, $\tau \in \boldsymbol{L}(\boldsymbol{M})$. Обратно, если некоторая цепочка переводит состояние q_s в состояние q_f, то либо она есть пустая цепочка ε, либо её несет путь, который, перейдя из q_s в q'_s и затем пройдя несколько раз по пути из q'_s в q'_f и вернувшись из q'_f в q'_s по ε-переходу, в конце концов из q'_f по ε-переходу завершается в q_f. Поэтому такая цепочка $\tau \in \boldsymbol{L}(\boldsymbol{M'})^*$.

Таким образом, класс конечно-автоматных языков содержит языки $\varnothing$, $\{\varepsilon\}$, $\{t\}$ для всех $t \in \boldsymbol{T}$ и замкнуто относительно операций объединения, конкатенации и итерации.

Автомат $\boldsymbol{M}$, построенный в доказательстве леммы II.3.2.1 по регулярному выражению ρ, не всегда является простым. Например:

1) при построении автомата для объединения $\boldsymbol{M'}$ и $\boldsymbol{M''}$ можно объединить их: начальные состояния в одно, если в них нет переходов

из других состояний; финальные состояния, если из них нет переходов в другие состояния и алфавиты $\boldsymbol{M'}$ и $\boldsymbol{M''}$ совпадают. Тогда не потребуются новое начальное состояние, новое заключительное состояние и ε-переходов.

2) при построении автомата для конкатенации $\boldsymbol{M'}$ и $\boldsymbol{M''}$, если из заключительного состояния $\boldsymbol{M'}$ нет переходов в другие состояния, то его можно объединить с начальным состоянием $\boldsymbol{M''}$.

3) для реализации цепочки $t_1t_2 \ldots t_n$, где $t_i \in \boldsymbol{T}$ (i=1,2, …, n), можно просто использовать автомат с (n+1) состоянием q_i (i=0,1,…, n) и командами $\delta(q_{i-1}, t_i)= q_i$, в котором нет пустых ε-переходов, участвующих в общей конструкции для конкатенации.

Примеры II.3.2. Применим лемму II.3.2.1 к регулярному выражению ρ = (1∨01∨001)*(ε∨0∨00), порождающее язык, состоящий из всех цепочек, которые не содержат подцепочки '000'.

На рисунке II.3.2.6.a) и II.3.2.6.b) представлены диаграммы автоматов $\boldsymbol{M_1}$ и $\boldsymbol{M}_2$, построенных по выражениям ρ_1 = (1∨01∨001) и ρ_2 = (ε∨0∨00), соответственно, с помощью конструкций для конкатенации и объединения.

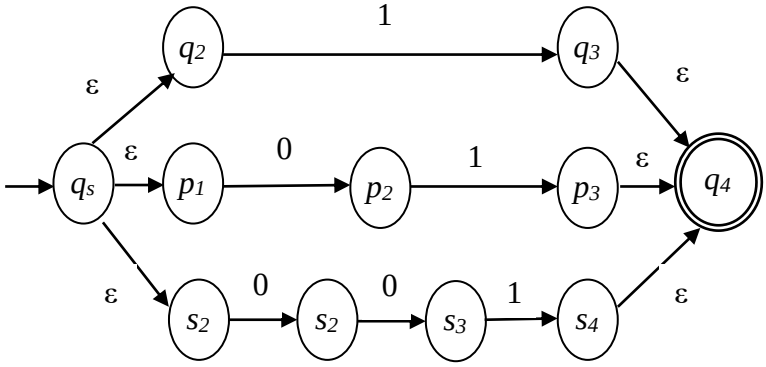


Рисунок II.3.2.6.a) КА $\boldsymbol{M_1}$

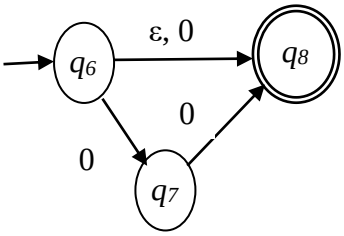


Рисунок II.3.2.6.b) КА $\boldsymbol{M_2}$

Как мы отмечали выше, автомат $\boldsymbol{M_1}$ можно было бы еще упростить, склеив начальные состояния q_2, p_1 и s_1, а также заключительные состояния q_3, p_3 и s_4.

Автомат $\boldsymbol{M}_3$ для регулярного выражения ρ'* = (1∨01∨001)* получается из автомата $\boldsymbol{M}_1$ добавлением нового начального состояния

q_0 и заключительного состояния q_5, а такжк ε-переходов из q_0 в q_1 и q_5, из q_4 в q_5 и q_1. Затем результирующий автомат $\boldsymbol{M}_4$ для исходного РВ ρ получается последовательным соединением $\boldsymbol{M}_3$ и $\boldsymbol{M}_2$, это показано на рисунке II.3.2.7.

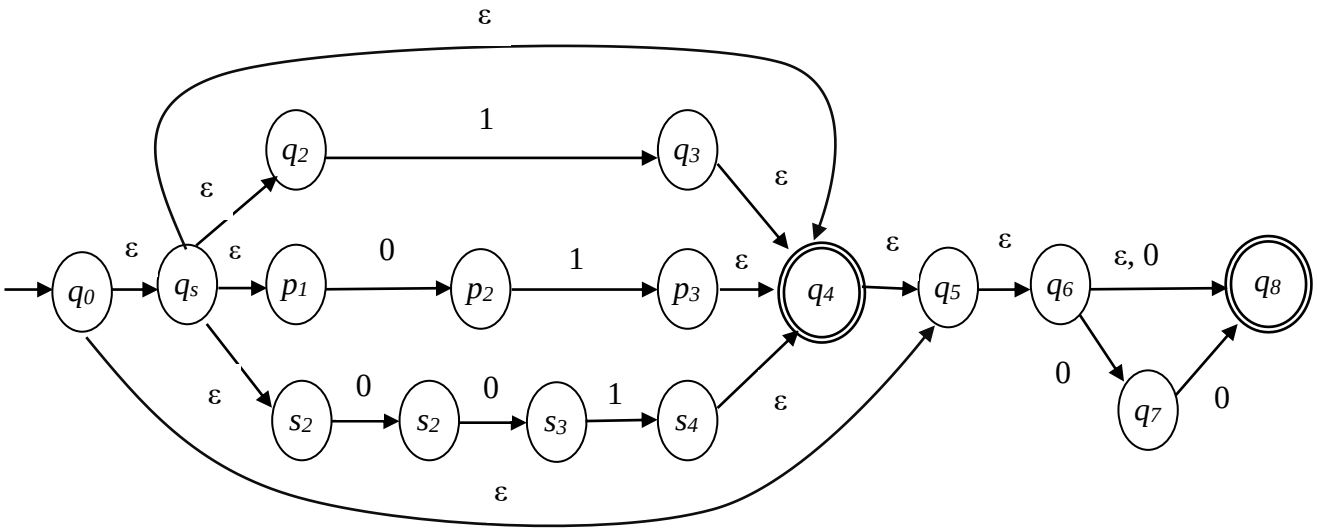


Рисунок II.3.2.7. Результирующий автомат $\boldsymbol{M}_4$

Следствие II.3.2.1. Для каждого РВ можно эффективно построить детерминированный конечный автомат (ДКА), который распознает язык, представляемый этим РВ.

Лемма II.3.2.2. Для любого НКА $\boldsymbol{M}$, распознающего язык $\boldsymbol{L}(\boldsymbol{M})$, можно построить соответствующее РВ ρ, которое порождает язык $\boldsymbol{L}(\rho)$, такой, что $\boldsymbol{L}(\rho) \subseteq \boldsymbol{L}(\boldsymbol{M})$.

Доказательство. Пусть задан $\boldsymbol{L}(\boldsymbol{M})$ – язык, распознаваемый ДКА $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_1, \boldsymbol{F}, \vdash, \dashv, \delta>$. Введем расширенную функцию переходов ДКА $\boldsymbol{M}$: $\delta^{\varepsilon}(q, \tau) = p$, где $\tau \in \boldsymbol{T}^*$, q, $p \in \boldsymbol{Q}$ тогда и только тогда, когда $(q, \tau) \models^* (p, \varepsilon)$.

Обозначим через ρ_{ij}^{k} множество всех цепочек x таких, что $\delta^{\varepsilon}(q_i, x) = q_j$ и если $\delta^{\varepsilon}(q_i, y) = q_m$ для любой цепочки y - префикса x, отличного от x и ε, то $m \leqslant k$. Иными словами, ρ_{ij}^{k} есть множество всех цепочек, которые переводят ДКА из состояния q_i в состояние q_j, не проходя ни через какое состояние q_m для $m > k$. Однако, i и j могут быть больше k.

ρ_{ij}^{k} может быть определено рекурсивно следующим образом:

$$\rho_{ij}^{m} = \{t:\ t \in \boldsymbol{T}\ \&\ \delta(q_i, t) = q_j\},$$

$$\rho_{ij}^{k} = \rho_{ij}^{k-1} \cup \rho_{ik}^{k-1}(\rho_{kk}^{k-1})^{*}\rho_{kj}^{k-1}, \text{ где } 1 \leqslant k \leqslant n.$$

Таким образом, определение ρ_{ij}^{k} означает, что для входной цепочки τ, переводящей ДКА $\boldsymbol{M}$ из q_i в q_j без перехода через состояния с номерами, большими k, справедливо ровно одно из следующих двух утверждений:

1) Цепочка τ входит ρ_{ij}^{k-1}, т.е. при анализе цепочки τ ДКА никогда не достигает состояний с номерами, большими/равными k;

2) Цепочка τ может быть представлена в виде $\tau = \tau_1\tau_2\tau_3$, где подцепочка $\tau_1 \in \rho_{ik}^{k-1}$ переводит ДКА в состояние q_k; подцепочка $\tau_2 \in (\rho_{kk}^{k-1})^{*}$ переводит ДКА из состояния q_k обратно в состояние q_k, не проходя через состояния с номерами, большими / равными k; подцепочка $\tau_3 \in \rho_{kj}^{k-1}$ переводит ДКА из состояния q_k в состояние q_j. Диаграмма этого ДКА представлена на рисунке II.3.2.8.

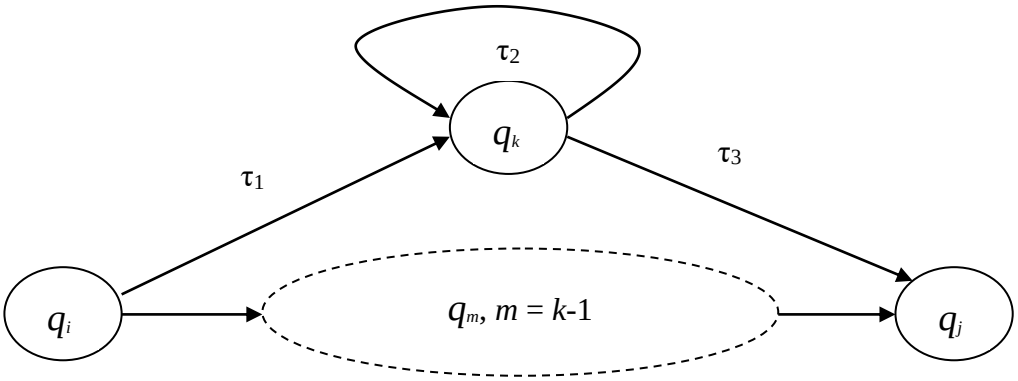


Рисунок II.3.2.8. ДКА $\boldsymbol{M}$ для РВ ρ

Тогда $\boldsymbol{L}(\rho) = \bigcup_{q_j \in F} \rho_{1j}^{k}$. Индукцией по k можно показать, что это множество является регулярным языком (множеством).

Из леммы II.3.2.1 и леммы II.3.2.2 следует, что для всякого регулярного множества имеется конечный автомат, распознающий в точности это регулярное множество, и наоборот - язык, распознаваемый конечным автоматом есть регулярное множество, т.е. справедлива следующая теорема.

Теорема II.3.2.1. Язык является регулярным множеством тогда и только тогда, когда он распознается конечным автоматом.

Таким образом, можно сделать вывод, что класс языков, распознаваемых конечными автоматами совпадает с классом регулярных языков, порождаемых регулярными выражениями. Их можно назвать просто автоматными языками или регулярными языками. Этот факт широко используется при разработке различных языковых процессоров.

Задания II.3.2. Построить конечные автоматы, эквивалентные регулярным выражениям:

1. $(a \vee b \vee ab)^*$.
2. $(a^*b)^* \vee (b^*a)^*$.
3. $(ba \vee a^*ab)^*$.
4. $(b^+a)^*(b \vee a)b^*$.
5. $(b^*a)b \vee (ab^*)a$.
6. $(ba^*) \vee (b \vee a)b^*$.
7. $(a^* \vee b)b \vee (a \vee b^*)a$.
8. $(ba)^*b \vee a\ b^*$.

Вопросы II.3.2:

1. Для двух регулярных выражений, порождающих одно и то же регулярный язык, найдется ли конечный автомат, который будет эквивалентным этим регулярным выражениям?

3. Имеется ли регулярное выражение, эквивалентное конечному автомату с параметрами $\boldsymbol{Q}$ ={1, 2}, $\boldsymbol{T}$={a, b}, $\boldsymbol{I}$={1}, $\boldsymbol{F}$ ={2}, Δ = {<1, aaa, 1>, <1, ab, 2>,<1, b, 2>, <2, ε, 1>}?

4. Равны ли конечные автоматы, эквивалентные регулярным выражениям $(a \vee b)^*$ и $(a \vee b \vee ab)^*$?

5. Равны ли конечные автоматы, эквивалентные регулярным выражениям $(ab \vee b)$ $(a \vee ba)$ и $(a \vee b)^*$ $(b \vee a)^*$?

6. Равны ли конечные автоматы, эквивалентные регулярным выражениям $(a \vee b)^*(b \vee a)^*$ и $(a \vee b \vee ab)^*$?

7. Имеется ли регулярное выражение, эквивалентное недетерминированному конечному автомату со следующими параметрами $\boldsymbol{Q}$ = {A, B, C, D, E, F, G}, $\boldsymbol{T}$ = {a,b,c}, $\boldsymbol{I}$=A, $\boldsymbol{F}$ = {D},

$\delta(A, b)=\{D\}$, $\delta(D, c)=\{F\}$, $\delta(F, a)=\{D\}$, $\delta(F, b)=\{F\}$, $\delta(E, c)=\{C\}$,

$\delta(C, b)=\{C\}$,$\delta(A, c)=\{B\}$, $\delta(A, a)=\{A\}$, $\delta(F, c)=\{C\}$, $\delta(E, a)=\{F\}$, $\delta(F, a)=\{E\}$, $\delta(C, a)=\{C\}$,$\delta(F, a)=\{B\}$, $\delta(B, b)=\{A\}$, $\delta(E, c)=\{D\}$.

II.3.3. Эквивалентность праволинейных грамматик и конечных автоматов

В этом параграфе будут показываться, что класс праволинейных языков, порождаемых праволинейными грамматиками, в точности совпадает с классом языков, распознаваемых конечными автоматами, а также предложены примеры, даны задания, сформулированы вопросы [1,3,4,11,13,14,17,20,24,26,28,34].

Это доказывается по лемме II.3.3.1 построением недетерминированного конечного автомата (НКА), соответствующего заданной праволинейной грамматике (ПЛГ), и наоборот, лемме *II*.3.3.2 построением ПЛГ, соответствующего НКА.

Лемма II.3.3.1. Для любой ПЛГ $\boldsymbol{G}$, порождающей праволинейный язык $\boldsymbol{L}(\boldsymbol{G})$, можно построить соответствующий КА $\boldsymbol{M}$, распознающий язык $\boldsymbol{L}(\boldsymbol{M})$, такой, что $\boldsymbol{L}(\boldsymbol{M})\subseteq\boldsymbol{L}(\boldsymbol{G})$.

Доказательство. Пусть задана ПЛГ $\boldsymbol{G} = <\boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{P}, S>$. Без ограничения общности можно предположить, что исходный язык $\boldsymbol{L}(\boldsymbol{G})$ порожден ПЛГ, не содержащей правил вида $A\to\alpha$, где $\alpha\in \boldsymbol{T}^{+}$. Тогда можно построить КА $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \vdash, \dashv, \delta>$ следующим образом:

– состояниями $\boldsymbol{M}$ будут нетерминалы $\boldsymbol{G}$ плюс новое состояние p, не принадлежащее $\boldsymbol{N}$, так что $\boldsymbol{Q} = \boldsymbol{N}\cup\{p\}$;

– в качестве начального состояния $\boldsymbol{M}$ примем начальный нетерминал S, т.е. $q_s = S$;

– если $\boldsymbol{P}$ содержит правило $S\to\varepsilon$, то $\boldsymbol{F}=\{S, p\}$, $A\in\boldsymbol{N}$, иначе $\boldsymbol{F}=\{p\}$;

– если $\boldsymbol{P}$ содержит правило $q\to tB$, где $t\in\boldsymbol{T}$, $B\in\boldsymbol{N}$, то $\delta(q,t)=B$ для ДКА или $\delta(q, t)\ni B$ для НКА.

КА $\boldsymbol{M}$, читая вход τ, моделирует вывод τ в грамматике $\boldsymbol{G}$. Покажем, что $\boldsymbol{L}(\boldsymbol{M})\subseteq\boldsymbol{L}(\boldsymbol{G})$.

Пусть $\tau = t_1t_2...t_n \in\boldsymbol{L}(\boldsymbol{G})$, $n \geqslant 1$. Тогда в грамматике $\boldsymbol{G}$ имеются выводы $S\Rightarrow t_1A_1\Rightarrow t_1t_2A_2\Rightarrow\ldots\Rightarrow t_1t_2...t_{n-1}A_{n-1}\Rightarrow t_1t_2...t_n$ для некоторой последовательности нетерминалов A_1, A_2,..., A_{n-1}. По определению, $\delta(S, t_1)$ содержит A_1, $\delta(A_1, t_2)$ содержит A_2, и т.д. $\delta(A_{n-2}, t_{n-1})$ содержит A_{n-}

$_1$, $\delta(A_{n-1}, t_n)$ содержит p. Так что $\tau \in \boldsymbol{L}(\boldsymbol{G})$, поскольку $\delta(S, \tau)$ содержит p, а $p \in \boldsymbol{F}$. Если $\varepsilon \in \boldsymbol{L}(\boldsymbol{G})$, то $S \in \boldsymbol{F}$, так что $\varepsilon \in \boldsymbol{L}(\boldsymbol{G})$.

Аналогично, если цепочка $\tau = t_1 t_2 ... t_n \in \boldsymbol{L}(\boldsymbol{G})$, $n \geqslant 1$, то существует последовательность состояний S, A_1, A_2, ..., A_{n-1}, p такая, что $\delta(S, t_1)$ содержит A_1, $\delta(A_1, t_2)$ содержит A_2, и т.д. $\delta(A_{n-2}, t_{n-1})$ содержит A_{n-1}, $\delta(A_{n-1}, t_n)$ содержит p, т.е. в грамматике $\boldsymbol{G}$ имеются выводы

$$S \Rightarrow t_1 A_1 \Rightarrow t_1 t_2 A_2 \Rightarrow \ldots \Rightarrow t_1 t_2 ... t_{n-1} A_{n-1} \Rightarrow t_1 t_2 ... t_n$$

вывод в $\boldsymbol{G}$ и $\tau \in \boldsymbol{L}(\boldsymbol{G})$. Если $\varepsilon \in \boldsymbol{L}(\boldsymbol{M})$, то $S \in \boldsymbol{F}$, так что $S \to \varepsilon \in \boldsymbol{P}$ и $\varepsilon \in \boldsymbol{L}(\boldsymbol{G})$.

Лемма II.3.3.2. Для любого КА $\boldsymbol{M}$, распознающего язык $\boldsymbol{L}(\boldsymbol{M})$, можно построить соответствующую ПЛГ $\boldsymbol{G}$, порождающую праволинейный язык $\boldsymbol{L}(\boldsymbol{G})$, такой, что $\boldsymbol{L}(\boldsymbol{G}) \subseteq \boldsymbol{L}(\boldsymbol{M})$.

Доказательство. Пусть задан КА $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \vdash, \dashv, \delta>$. Пусть $q \in \boldsymbol{Q}$, $p \in \boldsymbol{Q}$, $t \in \boldsymbol{T}$, $\tau \in \boldsymbol{T}^*$, тогда грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ строится так:

1) нетерминалами грамматики $\boldsymbol{G}$ будут состояния КА $\boldsymbol{M}$, т.е. $\boldsymbol{N} = \boldsymbol{Q}$;

2) начальным нетерминалом грамматики $\boldsymbol{G}$ будет начальное состояние КА $\boldsymbol{M}$, т.е. $S = q_s$;

3) $q \to tp \in \boldsymbol{P}$ для некоторого $p \in \boldsymbol{Q}$, если $\delta(q, t) = p$ в случае детерминированного КА или $\delta(q, t) \ni p$ в случае недетерминированного КА, т.е. $(q, t\tau) \models (p, \tau)$, и если $p \in \boldsymbol{F}$, то $\tau = \varepsilon$;

4) $S \to \varepsilon \in \boldsymbol{P}$, если $q_s \in \boldsymbol{F}$.

Заметим, что если $\boldsymbol{P}$ содержит правило $S \to \varepsilon$, то S не встречается в правых частях правил и $\boldsymbol{F} = \{S, p\}$, иначе $\boldsymbol{F} = \{p\}$.

Можно показать, что каждый шаг вывода в грамматике $\boldsymbol{G}$ имитирует такт автомата $\boldsymbol{M}$. Докажем индукцией по i, что $q \Rightarrow^{i+1} \tau$ для $q \in \boldsymbol{Q}$ тогда и только тогда, когда $(q, \tau) \models^{i} (p, \varepsilon)$ для некоторого $p \in \boldsymbol{F}$.

Базис индукции. Для $i=0$ очевидно, что $q \Rightarrow \varepsilon$ тогда и только тогда, когда $(q, \varepsilon) \models^{0} (q, \varepsilon)$ для $q \in \boldsymbol{F}$.

Шаг индукции. Предположим, что $q \Rightarrow^{i+1} \tau$ истинно для i, и возьмем $\tau = tx$, где $|x| = i$. Тогда $q \Rightarrow^{i+1} \tau$ равнозначно тому, что имеет место $q \Rightarrow tp \Rightarrow^{i} tx$ для некоторого $p \in \boldsymbol{Q}$. Но $q \Rightarrow tp$ равнозначно

$\delta(q,t)=p$. По предположению индукции $p \Rightarrow^{i} x$ тогда и только тогда, когда $(p,x) \models^{i-1} (r,\varepsilon)$ для некоторого $r \in \boldsymbol{F}$. Следовательно, $q \Rightarrow^{i+1} \tau$ равнозначно $(q,\tau) \models^{i} (r, \varepsilon)$ для некоторого $r \in \boldsymbol{F}$. Итак, утверждение $q \Rightarrow^{i+1} \tau$ истинно для всех $i \geq 0$.

Отсюда заключаем, что $q_0 \Rightarrow^{+} \tau$ тогда и только тогда, когда $(q_0,\tau) \models^{*} (r, \varepsilon)$ для некоторого $r \in \boldsymbol{F}$. Таким образом, $\boldsymbol{L}(\boldsymbol{G}) \subseteq \boldsymbol{L}(\boldsymbol{M})$.

Из леммы *II*.3.3.1 и леммы *II*.3.3.2 следует, что для всякого праволинейного языка имеется конечный автомат, распознающий в точности это праволинейный язык, и наоборот - язык, распознаваемый конечным автоматом есть праволинейный язык, т.е. справедлива следующая теорема.

Теорема II.3.3.1. *Я*зык является праволинейным языком тогда и только тогда, когда он распознается КА.

Таким образом, класс языков, распознаваемых конечными автоматами совпадает с классом праволинейных языков.

Примеры II.3.3. Пусть задана праволинейная грамматика $\boldsymbol{G} = \langle \boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S \rangle$ с нетерминальными символами $\boldsymbol{N} = \{S, A\}$, терминальными символами $\boldsymbol{T} = \{0,1\}$ и правилами вывода $\boldsymbol{P} = S \rightarrow 00, S \rightarrow A, A \rightarrow 10A, A \rightarrow 0$. *Я*зык, порождаемый этой грамматикой, распознается конечным автоматом $\boldsymbol{M} = \langle \boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \vdash, \dashv, \delta \rangle$ (или в терминах отношения $\boldsymbol{M} = \langle \boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \vdash, \dashv, \Delta \rangle$), где $\boldsymbol{Q} = \{S, A, B\}$, $q_s = \{S\}$, $\boldsymbol{F} = \{B\}$ и $\delta(S, 00)=B$, $\delta(S, \varepsilon)=A$, $\delta(A, 10)=A$, $\delta(A, 0)=B$ (или в терминах отношения $\Delta = \{\langle S, 00, B \rangle, \langle S, \varepsilon, A \rangle, \langle A, 10, A \rangle, \langle A, 0, B \rangle\}$). Диаграмма этого автомата показана на рисунке II.3.3.

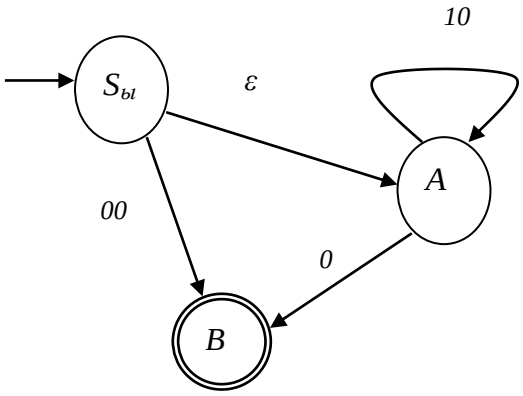


Рисунок II.3.3. Диаграмма НКА, эквивалентный ПЛГ.

Задания II.3.3. Построить конечные автоматы, эквивалентные праволинейным грамматикам:

1. $\boldsymbol{N}=\{S, A\}$, $\boldsymbol{T}=\{a, b\}$, $\boldsymbol{P}=\{S\to aA, A\to aA, A\to b\}$;
2. $\boldsymbol{N}=\{S, A\}$, $\boldsymbol{T}=\{a, b\}$, $\boldsymbol{P}=\{S\to Aa, A\to Aa, A\to b\}$;
3. $\boldsymbol{N}=\{S\}$, $\boldsymbol{T}=\{a\}$, $\boldsymbol{P}=\{S\to a, S\to aaS\}$;
4. $\boldsymbol{N}=\{S\}$, $\boldsymbol{T}=\{a\}$, $\boldsymbol{P}=\{S\to a, S\to Saa\}$;
5. $\boldsymbol{N}=\{S, A\}$, $\boldsymbol{T}=\{a, b\}$, $\boldsymbol{P}=\{S\to 0A|1S|\varepsilon, A\to 0B|1A, B\to 0S|1B\}$;
6. $\boldsymbol{N}=\{A,B, D\}$, $\boldsymbol{T}=\{a, b, d\}$, $\boldsymbol{S}=A$, $\boldsymbol{P}=\{A\to aB, B\to aB, B\to b, B\to bD, D\to d, D\to dD, A\to aD, A\to a\}$;
7. $\boldsymbol{N}=\{S,E,F\}$, $\boldsymbol{T}=\{a,b\}$, $\boldsymbol{P}=\{S\to aaE, S\to F, F\to baF, F\to aE, E\to\varepsilon\}$.

Вопросы II.3.3:

1. Для двух праволинейных грамматик, порождающих один и то же язык, найдется ли конечный автомат, который будет эквивалентным обеим праволинейным грамматикам?

2. Имеется ли праволинейная грамматика, эквивалентная конечному автомату $\boldsymbol{Q}=\{1, 2\}$, $\mathbf{T}=\{a, b\}$, $\boldsymbol{I}=\{1\}$, $\boldsymbol{F}=\{2\}$,

$$\Delta = \{<1, aaa, 1>, <1, ab, 2>, <1, b, 2>, <2, \varepsilon, 1>\}.$$

3. Равны ли конечные автоматы, эквивалентные праволинейным грамматика $\boldsymbol{N}=\{S\}$, $\boldsymbol{T}=\{a\}$, $P=\{S\to a, S\to aaS\}$ и $\boldsymbol{N}=\{S\}$, $\boldsymbol{T}=\{a\}$, $P=\{S\to a, S\to Saa\}$?

4. Будет ли конечным автоматом распознаватель, который распознает цепочки, порождаемые следующими правилами некоторой грамматики $\{a,b\}^* - (\{a^n : n\geq 0\} \square \{b^n : n\geq 0\}$?

5. Существует ли такая праволинейная грамматика $\boldsymbol{G}$, что язык $\boldsymbol{L}(\boldsymbol{G})^R$ – обращение языка $\boldsymbol{L}(\boldsymbol{G})$ не порождается ни одной праволинейной грамматикой, имеющей столько же правил, сколько грамматика $\boldsymbol{G}$?

6. Существует ли такая праволинейная грамматика G, что язык $\boldsymbol{L}(\boldsymbol{G})^R$ не порождается ни одной праволинейной грамматикой с количеством правил $n+1$, где n - количество правил в грамматике $\boldsymbol{G}$?

7. Распознает ли КА язык, порождаемый ПЛГ S→Sb, S→b?

II.3.4. Алгоритмические проблемы регулярных языков

В этом параграфе будут обсуждены алгоритмические проблемы регулярных языков, предложены примеры, даны задания, сформулированы вопросы [1,3,4,11,13,14,17,20,24,26,28,34].

Рассмотрим следующие алгоримически разрешимые проблемы:

1. *Проблема замкнутности*: “При применении множественной операции к языкам определенного типа выяснить, будет ли результат иметь такой же тип?”.

2. *Проблема принадлежности*: “Принадлежит ли заданная цепочка ω заданному регулярному языку?”.

3. *Проблема пустоты*: “Для произвольного регулярного языка ***L***требуется выяснить, пуст ли регулярный язык ***L***, т.е. $L=\varnothing$?”.

4. *Проблема пустоты пересечения*: “По произвольным двум регулярным языкам L_1 и L_2 в алфавитие ***T*** требуется определить, пусто ли их пересечение, т.е. $L_1 \cap L_2 = \varnothing$?”.

5. *Проблема включения*: “По произвольным двум регулярным языкам L_1 и L_2 в алфавитие ***T*** требуется определить первый язык включается ли во второй язык, т.е. $L_1 \subseteq L_2 = \varnothing$?”.

6. *Проблема эквивалентности*: “Определяют ли заданные два способа описания регулярных языков один и тот же язык?”.

7. *Проблема бесконечности*: “Для заданной грамматики ***G*** выяснить будет ли $L(G)$ бесконечным языком, т.е. $L(G) = \infty$?”.

Поскольку все способы описания регулярных языков (регулярные выражения, линейные грамматики, конечные автоматы и др.) эквивалентны между собой, то решения перечисленных алгоритмических проблем достаточно показать только для одного из этих способов.

Дадим алгоритмы, которые решают упомянутые выше проблемы в том случае, когда регулярные языки определяются с помощью конечных автоматов.

Проблему замкнутности регулярных языков можно решить с помощью следующего алгоритма.

Алгоритм II.3.4.1. Решение проблемы замкнутности для праволинейных грамматик.

Вход. Праволинейные грамматики

$G_1 = \langle N_1, T_1, P_1, S_1 \rangle$ и $G_2 = \langle N_2, T_2, P_2, S_2 \rangle$, $N_1 \cap N_2 = \varnothing$.

Выход. "ДА", если $L(G_1) \cup L(G_2)$ порождается грамматикой

$G = \langle N_1 \cup N_2 \cup \{S\}, T, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}, S \rangle$,

где $S \notin N_1 \cup N_2 \cup T$.

"НЕТ" в противном случае.

Метод. Получить выводы, выводимые из S. Если результатом является цепочка терминальных из T, то сказать "ДА", в противном случае сказать "НЕТ".

Решение проблемы принадлежности для конечных автоматов решается с помощью следующего алгоритма.

Алгоритм II.3.4.2. Решение проблемы принадлежности для конечных автоматов.

Вход. Конечный автомат $M = \langle Q, T, q_0, F, \delta \rangle$ и цепочка $\omega \in T^*$.

Выход. "ДА", если $\omega \in L(M)$; "НЕТ", если $\omega \notin L(M)$.

Метод. Пусть $\omega = a_1 a_2 ... a_n$. Найти последовательно состояния $q_1 = \delta(q_0, a_1)$, $q_2 = \delta(q_1, a_2)$,... $q_n = \delta(q_{n-1}, a_n)$. Если $q_n \in F$, сказать "ДА"; если $q_n \notin F$, сказать „НЕТ".

Проблема пустоты регулярных языков решается с помощью следующего алгоритма.

Алгоритм II.3.4.3. Решение проблемы пустоты для конечных автоматов.

Вход. Конечный автомат $M = \langle Q, T, q_0, F, \delta \rangle$.

Выход. "ДА", если $L(M) \neq \varnothing$, "НЕТ" в противном случае.

Метод. Вычислить множество состояний, достижимых из q_0. Если это множество содержит какое-нибудь заключительное состояние, то сказать "ДА", а в противном случае сказать "НЕТ".

Решение проблемы пустоты пересечения двух регулярных языков определяется следующей теоремой.

Теорема II.3.4.1. Проблема пустоты пересечения для класса регулярных языков алгоритмически разрешима.

Доказательство. Если регулярные языки L_1 и L_2 распознаются детерминированными конечными автоматами M_1 и M_2, то можно построить конечный автомат M, рапознающий язык $L = L_1 \cap L_2$ и решить для языка Lпроблему пустоты.

Решение проблемы включения для двух регулярных языков определяется следующей теоремой.

Теорема II.3.4.2. Проблема включения для регулярных языков алгоритмически разрешима.

Доказательство. Разрешимость проблемы включения для регулярных языков вытекает из следующей эквивалентности

$$L_1 \subseteq L_2 \Leftrightarrow L_1 \cap (T^* \setminus L_2) = \varnothing$$

Решение проблемы эквивалентности двух регулярных языков устанавливается следующим алгоритмом.

Алгоритм II.3.4.4. Решение проблемы эквивалентности для конечных автоматов.

Вход. Два конечных автомата $M_1 = \langle Q_1, T_1, q_1, F_1, \delta_1 \rangle$ и $M_2 = \langle Q_2, \delta_2, q_2, F_2, T_2 \rangle$, таких, что $Q_1 \cap Q_2 = \varnothing$.

Выход. “ДА”, если $L(M_1) = L(M_2)$, “НЕТ” в противном случае.

Метод. Построить конечный автомат

$$M = \langle Q_1 \cup Q_2, T_1 \cup T_2, q_1, F_1 \cup F_2, \delta_1 \cup \delta_2 \rangle$$

Определим различимы ли автоматные состояния q_1 и q_2. Если раличаются, то сказать “НЕТ”, в противном случае сказать “ДА”.

Для решения проблемы эквивалентности можно воспользоваться алгоритмом II.3.4.2, так как $L(M_1) = L(M_2)$ тогда и только тогда, когда $(L(M_1) \cap L(M_2)) \cup (L(M_2) \cap L(M_1)) = \varnothing$ или воспользоваться следующей эквивалентностью:

$$L_1 = L_2 \Leftrightarrow (L_1 \subseteq L_2) \,\&\, (L_2 \subseteq L_1)$$

Теперь покажем решение проблемы бесконечности конечно-автоматного языка. Для этого доказывается следующая теорема.

Теорема II.3.4.3. Множество цепочек, распознаваемых конечным автоматом $\boldsymbol{M}$ с n состояниями, бесконечно тогда и только тогда, когда он содержит цепочку длиной l, $n \le l < 2n$.

Доказательство.

Необходимость доказывается способом от противного. Пусть конечный автомат $\boldsymbol{M}$ распознает бесконечное множество цепочек, и ни одна из них не имеет длину l, $n \le l < 2n$. Если бы в множестве $\boldsymbol{L}(\boldsymbol{M})$ существовали только цепочки длиной $l<n$, то 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 $x \in \boldsymbol{L}(\boldsymbol{M})$ и $|x| \ge 2n$. Очевидно, что существует такое состояние $q \in \boldsymbol{Q}$, 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 $x_1x_3 \in \boldsymbol{L}(\boldsymbol{M})$, 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 $\boldsymbol{L}(\boldsymbol{M})$ цепочек длиной $n \le l < 2n$ не существует, то $|x_1x_3| \ge 2n$. Следовательно, вопреки предположению, что $x = x_1x_2x_3 \in \boldsymbol{L}(\boldsymbol{M})$ - одна из самых коротких цепочек, длина которой больше или равна $2n$, нашлась более короткая цепочка $x_1x_3 \in \boldsymbol{L}(\boldsymbol{M})$ и тоже с длиной, большей или равной $2n$. Это противоречие доказывает необходимость.

Достаточность вытекает из следующих рассуждений. Пусть существует $x \in \boldsymbol{L}(\boldsymbol{M})$, причем $n \le |x| < 2n$. Как и ранее, утверждаем, что существует $q \in \boldsymbol{Q}$, $x = x_1x_2x_3$, где $x_2 \ne \varepsilon$, и $\delta(q_0, x_1) = q$, $\delta(q, x_2) = q$, $\delta(q, x_3) \in F$. Но тогда цепочки вида $x_1x_2^ix_3 \in \boldsymbol{L}(\boldsymbol{M})$ при любом i. Очевидно, что множество $\boldsymbol{L}(\boldsymbol{M})$ бесконечно. Что и требовалось доказать.

Регулярное выражение можно превратить в эквивалентный конечный автомат с помощью алгоритма. Затем к полученному автомату можно применить один из алгоритмов III.4.2.1 - III.4.2.3.

В заключении можно жоказать следующую теорему.

Теорема II.3.4.4. Если множества определяются конечными автоматами, регулярными выражениями или праволинейными

грамматиками, то проблемы замкнутости, принадлежности, пустоты, эквивалентности и бесконечности для регулярных языков алгоритмически разрешимы.

Примеры II.3.4.

Пусть $\boldsymbol{T} = \{a, b, c\}$. Язык $\boldsymbol{L} = \boldsymbol{T}^* \setminus \{a^n b^n c^n: \ n>0\}$ является праволинейным, поскольку $\boldsymbol{L} = \boldsymbol{L}_1 \cup \boldsymbol{L}_2 \cup (\boldsymbol{T}^* \setminus \boldsymbol{L}_3)$, где языки $\boldsymbol{L}_1 = \{a^m b^n c^k: \ m \geq n, \ k>0\}$ и $\boldsymbol{L}_2 = \{a^m b^n c^k: \ n \geq k, \ m>0\}$ являются праволинейными, а язык $\boldsymbol{L}_3 = \{a^m b^n c^k: \ m>0, \ n>0, \ k>0\}$ является автоматным.

Задания II.3.4:

1. Для двух линейных грамматик $\boldsymbol{G}_1$ и $\boldsymbol{G}_2$, порождающие регулярные языки $\boldsymbol{L}(\boldsymbol{G}_1) = \{a, b\}^*$ и $\boldsymbol{L}(\boldsymbol{G}_2) = \{a, b, c\}^*$, вычислить значение выражения $\boldsymbol{L}(\boldsymbol{G}_1) \cap \boldsymbol{L}(\boldsymbol{G}_2)$.

2. Для регулярного языка $\{\gamma\delta: \ \gamma \in \{a, b\}^*, \ \delta \in \{a, b\}^*\}$ найти его порождающую линейную грамматику и распознающий конечный автомат.

3. Для регулярного языка $\{a\xi b: \ \xi \in \{a, b\}^*\} \cup \{b\xi a: \ \xi \in \{a, b\}^*\}$ найти порождающие его регулярное выражение и праволинейную грамматику.

4. Для регулярного языка $\{\xi b: \ \xi \in b\{a, b\}^*\} \cup \{a\xi: \ \xi \in a\{a, b\}^*\}$ найти порождающее его регулярное выражение и распознающий конечный автомат.

Вопросы II.3.4:

1. Существует ли алгоритм, который может выяснить, принадлежит ли заданная цепочка регулярному языку, порожденному праволинейной грамматикой?

2. Существует ли алгоритм, который выясняет пустототу регулярного языка, распознаваемого конечным автоматом?

3. Существует ли алгоритм, который выясняет, генерируют ли два регулярных выражения один и тот же регулярный язык?

4. Существует ли алгоритм, который выясняет, распознает ли два конечного автомата один и тот же регулярный язык?

III. БЕСКОНТЕКСТНЫЕ ЯЗЫКИ

III.1. Порождающие механизмы бесконтекстных языков

III.1.1. Бесконтекстные грамматики

В этой части рассматриваются бесконтекстные (контекстно-свободные) языки, даются механизмы их порождения и распознавания, показывается эквивалентность бесконтекстных грамматик и стековых автоматов, обсуждаются алгоритмические проблемы бесконтекстных языков, будут предложены примеры, даны задания, сформулированы вопросы [1-9,11-13,15-22,24,25, 27-32].

Определение III.1.1.1. *Я*зыки, порождаемые бесконтекстными грамматиками, называются бесконтекстными языками.

Напомним, что бесконтекстными (контекстно-свободными) грамматиками являются грамматики, в которых все правила вывода имеют вид $A\to\alpha$, где $A\in\boldsymbol{N}$, $\alpha\in(\boldsymbol{T}\cup\boldsymbol{N})^*$, т.е. нетерминал A заменяется цепочкой α во множестве терминалов и нетерминалов независимо от контекста, в котором встречается A (**I.3.2.**).

Бесконтекстные грамматики (БГ) занимают важное место в теории языков и служат для задания синтаксической структуры порождаемой цепочки с помощью последовательности применения правил вывода.

Примеры III.1.1.1. Пусть $\boldsymbol{G_1}$ = <$\boldsymbol{T}$, $\boldsymbol{N}$, $\boldsymbol{P}$, S>, где $\boldsymbol{N}$={S,A,B}, $\boldsymbol{T}$={a,b}, $\boldsymbol{P}$ = {$S\to aB$, $S\to bA$, $A\to aS$, $A\to bAA$, $B\to bS$, $A\to a$, $B\to aBB$, $B\to b$}.

Грамматика $\boldsymbol{G_1}$ является бесконтекстной, так как в каждом ее правиле вывода левая часть состоит из единственного нетерминала, а правая часть – из непустой цепочки терминалов и нетерминалов. В грамматике $\boldsymbol{G_1}$ типичные выводы таковы:

$S\Rightarrow aB\Rightarrow ab$,

$S\Rightarrow aB\Rightarrow abS\Rightarrow abbA\Rightarrow abba$,

$S\Rightarrow bA\Rightarrow ba$,

$S\Rightarrow bA\Rightarrow bbAA\Rightarrow bbaA\Rightarrow bbaa$.

Примененные правил вывода, которые порождают множество всех цепочек, состоящих из равного числа символов a и b.

В грамматике может быть несколько эквивалентных выводов, в которых применяются одни и те же правила в одних и тех же местах, но в различном порядке. Определить понятие эквивалентности двух выводов для грамматик произвольного вида сложно, но в условиях БГ можно ввести удобное графическое представление класса эквивалентных выводов, называемое деревом вывода.

Определение III.1.1.2. Помеченное упорядоченное дерево $\boldsymbol{D}$ называется деревом вывода (деревом разбора) в БГ $\boldsymbol{G}(S) = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$, если выполнены следующие условия:

(1) Корень дерева $\boldsymbol{D}$ (вершина, в которую не входит ни одна дуга) помечен S;

(2) Если $\boldsymbol{D}_1, \ldots, \boldsymbol{D}_k$ − поддеревья, над которыми доминируют прямые потомки корня дерева, и корень дерева $\boldsymbol{D}_i$ помечен X_i, то выражение $S \rightarrow X_1 X_2 \ldots X_k$ − правило из множества $\boldsymbol{P}$.

(3) Если X_i нетерминал, $\boldsymbol{D}_i$ состоит из единственной вершины, помеченной X_i и если X_i − терминал, то $\boldsymbol{D}_i$ для любого i=1, 2, …, k должно быть деревом вывода в грамматике $\boldsymbol{G}(X_i) = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, X_i>$,

(4) Если корень дерева имеет единственного потомка, помеченного ε, то этот потомок образует дерево, состоящее из единственной вершины, и выражение $S \rightarrow \varepsilon$ будет правилом из множества $\boldsymbol{P}$.

Таким образом, в дереве вывода каждая вершина помечается символом из множества $\boldsymbol{N} \cup \boldsymbol{T} \cup \{\varepsilon\}$. При этом, если внутренняя вершина дерева обозначена символом A, а его прямые потомки обозначаются символами $X_1, X_2, \ldots, X_n$, то выражение $A \rightarrow X_1 X_2 \ldots X_n$ будет правилом вывода грамматики.

Дерево вывода для БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$ может быть построено следующим образом:

1. Вершины дерева помечаются символами из множества $\boldsymbol{T} \cup \boldsymbol{N}$ в строго определенном порядке.

2. Если вершина с меткой X имеет хотя бы одну подчиненную вершину, то $X \in \boldsymbol{N}$. При этом корень дерева помечается $S \in \boldsymbol{N}$.

3. Если вершины X_1, X_2,…, X_k прямо подчинены вершине S, то правило $S \to X_1, X_2, \ldots, X_k$ должно принадлежать множеству $\boldsymbol{P}$.

Заметим, что существует естественное упорядочение вершин упорядоченного дерева, при котором прямые потомки вершины упорядочиваются “слева направо”.

Примеры III.1.1.2. На рисунке III.1.1.1 изображены деревья вывода в грамматике $\boldsymbol{G_2} = \boldsymbol{G}(S)$ с правилами $S \to aSbS|bSaS|\varepsilon$.

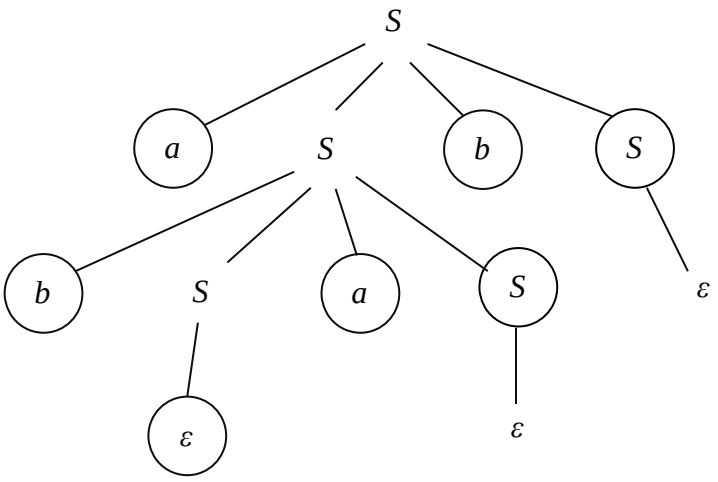


Рисунок III.1.1.1. Примеры дерева грамматики.

Пронумеруем вершины дерева сверху вниз и слева направо. Допустим, что X – вершина и X_1, ..., X_k – ее прямые потомки. Тогда для вершин X_i и X_j, если $i<j$, ($i=1,2, \ldots, k$, $j=1,2, \ldots, k$), то вершина X_i и все ее потомки считаются расположенными левее вершины X_j и всех ее потомков.

Пусть $\boldsymbol{D}$ − дерево вывода в БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$. Тогда можно ввести следующие новые понятия:

Определения III.1.1.3:

1. *Кроной* дерева вывода называется цепочка, которая получается, если выписать слева направо метки листьев;

2. *Сечением* дерева $\boldsymbol{D}$ называется такое множество $\boldsymbol{C}$ вершин дерева $\boldsymbol{D}$, что:

(1) никакие две вершины из C не лежат на одном пути в $\boldsymbol{D}$;

(2) ни одну вершину дерева $\boldsymbol{D}$ нельзя добавить к C, не нарушив свойства (1).

Можно показать, что деревья выводов представляют выводы в том смысле, что для каждого вывода выводимой цепочки α в БГ $\boldsymbol{G}$ можно построить дерево вывода в $\boldsymbol{G}$ с кроной α, и обратно.

Определение III.1.1.4. *Кроной сечения* дерева $\boldsymbol{D}$ называется цепочка, которая получается конкатенацией слева направо меток вершин, образующих некоторое сечение.

Примеры III.1.1.3. Кроной сечения дерева вывода, показанного на рисунке III.1.1.2, является цепочка *abSaSbS*.

Пусть $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$ − БГ. Тогда имеет место $S \Rightarrow^{*} \alpha$, когда в $\boldsymbol{G}$ существует дерево вывода $\boldsymbol{D}$ с кроной α.

Пусть $C_0, C_1, C_2, \ldots, C_n$ − такая последовательность сечений дерева $\boldsymbol{D}$, что:

(1) Сечение C_0 содержит только корень дерева $\boldsymbol{D}$;

(2) Сечение C_{i+1} для $0 \le i < n$ получается из сечения C_i заменой одной нетерминальной вершины ее прямыми потомками;

(3) C_n − крона дерева $\boldsymbol{D}$.

Если $S \Rightarrow^{*} \tau = \alpha_0, \alpha_1, \ldots, \alpha_n$ – левый вывод терминальной цепочки τ, то каждая α_i имеет вид $x_i A_i \beta_i$, где $x_i \in \boldsymbol{T}^{*}$, $A_i \in \boldsymbol{N}$ и $\beta_i \in (\boldsymbol{N} \cup \boldsymbol{T})^{*}$, $0 \le i < n$. В левом выводе каждая следующая цепочка вывода α_{i+1} получается заменой самого левого нетерминала A_i предыдущей цепочки α_i на правую часть некоторого правила. В правом выводе заменяется самый правый нетерминал.

Определения III.1.1.5*:*

1. Если сечение C_{i+1} получается из C_i заменой самой левой нетерминальной вершины в C_i ее прямыми потомками, то соответствующий вывод $\alpha_0, \alpha_1, \ldots, \alpha_n$ называется *левым выводом* цепочки α_n из α_0 в грамматике $\boldsymbol{G}$. *Правый вывод* определяется аналогично, надо только в предыдущем предложении читать "самой

правой" вместо "самой левой". Заметим, что по дереву вывода левый (или правый) вывод определяется однозначно.

2. Цепочка τ называется *левовыводимой* в грамматике $\boldsymbol{G}$, если существует левый вывод $S\Rightarrow^*\tau$, и пишется как $S\Rightarrow^*_{Gl}\tau$ (или $S\Rightarrow^*_{l}\tau$, когда ясно, какая грамматика $\boldsymbol{G}$ имеется в виду).

3. Цепочка τ называется *правовыводимой* в грамматике $\boldsymbol{G}$, если существует правый вывод $S\Rightarrow^*\tau$, и пишется $S\Rightarrow^*_{Gh}\tau$ (или $S\Rightarrow^*_{Gh}\tau$).

Таким образом один шаг левого вывода обозначается через $\Rightarrow_l$, а шаг правого вывода – через $\Rightarrow_h$.

Примеры III.1.1.4. Рассмотрим БГ $\boldsymbol{G}_a$ с правилами

$$E\to E+H\,/H,\ H\to H*F\,/F,\ F\to(E)\,/a$$

Дерево вывода, показанное на рисунке III.1.1.3, служит представлением двух эквивалентных выводов цепочки $a+a$:

1) левый вывод $E\Rightarrow E+H\Rightarrow H+H\Rightarrow F+H\Rightarrow a+H\Rightarrow a+F\Rightarrow a+a$,

2) правый вывод $E\Rightarrow E+H\Rightarrow E+F\Rightarrow E+a\Rightarrow H+a\Rightarrow F+a\Rightarrow a+a$.

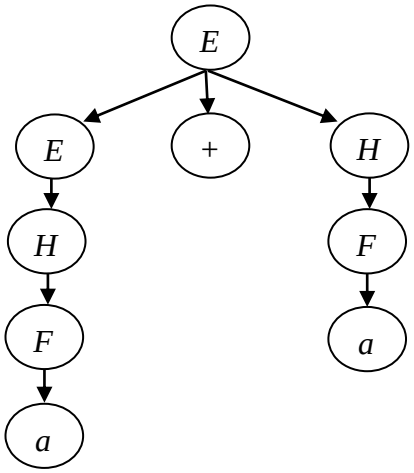


Рисунок. III.1.1.3. Дерево вывода цепочки $a+a$.

Определение III.1.1.6. БГ $\boldsymbol{G}$ называется *неоднозначной*, если существует хотя бы одна терминальная цепочка $\tau\in\boldsymbol{L}(\boldsymbol{G})$, которая является кроной двух или более различных деревьев выводов в $\boldsymbol{G}$. То есть, некоторая терминальная цепочка $\tau\in\boldsymbol{L}(\boldsymbol{G})$ имеет два или более разных левых (правых) вывода, В противном случае БГ $\boldsymbol{G}$ называется *однозначной*.

Примеры III.1.1.5. БГ G_a из примера III.1.1.5 является не однозначной, так как существует терминальная цепочка $a+a$ имеет два или более разных левых (правых) вывода:

левый вывод $E \Rightarrow E+H \Rightarrow H+H \Rightarrow F+H \Rightarrow a+H \Rightarrow a+F \Rightarrow a+a$,

правый вывод $E \Rightarrow E+H \Rightarrow E+F \Rightarrow E+a \Rightarrow H+a \Rightarrow F+a \Rightarrow a+a$.

Задания III.1.1:

1. Постройте дерево вывода для цепочки 10.1001 в БГ с правилами: $S \rightarrow S0|S1|D0|D1$, $D \rightarrow H$, $H \rightarrow 0|1|H0|H1$.
2. Постройте дерево вывода для цепочки *if* a then $b = a+b+b$ в БГ с правилами: $S \rightarrow$ if B then $S|B = E$, $E \rightarrow B|B+E$, $B \rightarrow a|b$
3. Постройте БГ, порождающую язык $\{a^{2n}b^m c^{2k} | m=n+k,\ m>1\}$, постройте дерево вывода и левосторонний вывод для *aabbbcccc*.
4. Постройте БГ, порождающую язык $\boldsymbol{L} = \{1^{3n+2}\ 0^n : n \geq 0\}$, постройте дерево вывода и левосторонний вывод для 1111111100.
5. Постройте в алфавите $\boldsymbol{T} = \{a, b\}$ языки $\boldsymbol{L}_1$ и $\boldsymbol{L}_2$, в которых буква b повторяется n раз $\boldsymbol{L}_1 = \{ab^n : n >= 0\}$, $\boldsymbol{L}_2 = \{b^n\ a : n >= 1\}$.
6. Постройте БГ, порождающую язык, который состоит из цепочек, начинающихся символом # и заканчивающихся символом !, между которыми расположена непустая цепочка из знаков+и –, не содержащая двух одинаковых символов, стоящих рядом.
7. Постройте БГ, порождающую правильные логические выражения с помощью конъюнкции & и дизъюнкция ∨, которые могут соединяться отношениями: >, <, =.

Вопросы III.1.1:

1. Как определяется левый (правый вывод) в БГ?
2. Какой тип языка $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n b^n c^n :\ n>1\}$?
3. Какой тип языка $\boldsymbol{L} = \{1^{3n+2}\ 0^n :\ n \geq 0\}$?
4. Какой тип грамматики, которая порождает множество $\{a \vee a^* a\}$?
5. Какой тип грамматики, которая порождает множество $\{a_1 a_2 \ldots a_n a_n \ldots a_2 a_1 :\ a_i \in \{0,1\},\ 1 \leq i \leq n\}$?
6. Какими будут левые выводы и правые выводы в БГ для этой цепочки 1111111100?
7.

III.1.2. Преобразование безконтекстных грамматик

В этом параграфе будут рассмотрены различные способы преобразования бесконтекстных грамматик, предложены примеры, даны задания, сформулированы вопросы [1-9,11-13,15-22,24,25, 27-32].

Иногда заданную грамматику часто требуется модифицировать так, чтобы порождаемый ею язык приобрел нужную структуру. Например, язык $\boldsymbol{L}(\boldsymbol{G}_a)$ может порождаться грамматикой $\boldsymbol{G}_a$ с правилами $E \to E+E \mid E*E \mid (E) \mid a$.

Но грамматика $\boldsymbol{G}_a$ имеет два недостатка:

Первый из них, из-за наличия правил $E \to E+E \mid E*E \mid (E) \mid a$ она неоднозначна. Эту неоднозначность можно устранить, взяв вместо $\boldsymbol{G}_a$ грамматику $\boldsymbol{G}_2$ с правилами $E \to E+H \mid E*H \mid H,\ H \to (E) \mid a$.

Другой недостаток грамматики $\boldsymbol{G}_a$, которым обладает также и $\boldsymbol{G}_1$, заключается в том, что операции+и * имеют один и тот же приоритет. Иначе говоря, структура выражений $a+a*a$ и $a*a+a$, которую придает им грамматика $\boldsymbol{G}_1$, подразумевает тот же порядок выполнения операций, что и в выражениях $(a \vee a)*a$ и $(a*a) \vee a$ соответственно.

Чтобы получить обычный приоритет операций+и *, при котором * предшествует $\vee$ и выражение $a \vee a*a$ понимается как $a \vee (a*a)$, надо перейти к грамматике $\boldsymbol{G}_a$.

Общего алгоритмического метода, который придавал бы заданному языку произвольную структуру, не существует. Но с помощью ряда преобразований можно видоизменить грамматику, не испортив порождаемого ею языка. Ниже мы рассмотрим несколько преобразований такого рода.

В некоторых случаях БГ может содержать бесполезные символы и правила. Например, в БГ $\boldsymbol{G} = \langle \{S, A\}, \{a, b\}, \boldsymbol{P}, S \rangle$, где $\boldsymbol{P} = \{S \to a, A \to b\}$, нетерминал A и терминал b не могут появиться ни в какой выводимой цепочке. Эти символы не имеют отношения к языку $\boldsymbol{L}(\boldsymbol{G})$, и их можно устранить из определения грамматики $\boldsymbol{G}$, не затронув языка $\boldsymbol{L}(\boldsymbol{G})$.

Определение III.1.2.1. Пусть $\gamma \in \boldsymbol{T}^*$, $\delta \in \boldsymbol{T}^*$, $\omega \in \boldsymbol{T}^*$ и $X \in \boldsymbol{N}$. Тогда нетерминальный символ X называется *полезным* (*useful*) в БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$, если он может участвовать в выводе вида $S \Rightarrow^* \gamma X \delta \Rightarrow^* \omega$, иначе он называется *бесполезным* (*useless*)

Чтобы установить, бесполезен ли нетерминал A, построим алгоритм, выясняющий, может ли нетерминал порождать какие-нибудь терминальные цепочки, т.е. мы построим алгоритм, решающий проблему пустоты множества $\{\tau: A \Rightarrow^* \tau, \tau \in \boldsymbol{T}^*\}$. Из существования такого алгоритма **следует** разрешимость проблемы пустоты для БГ.

Алгоритм III.1.2.1. Не пуст ли язык L(G)?

Вход. БГ $G = <T, N, P, S>$.

Выход. "ДА", если $\boldsymbol{L}(\boldsymbol{G}) \neq \varnothing$, "НЕТ" в противном случае.

Метод. Строим множества $\boldsymbol{N}_0$, $\boldsymbol{N}_1$ ... рекурсивно:

(1) Положить $\boldsymbol{N}_0 = \varnothing$ и $i=1$.

(2) Положить $\boldsymbol{N}_i = \{A \mid A \rightarrow \alpha \in \boldsymbol{P}$ и $\alpha \in <\boldsymbol{N}_{i-1} \cup \boldsymbol{T})^*\} \cup \boldsymbol{N}_{i-1}$.

(3) Если $\boldsymbol{N}_i = \boldsymbol{N}_{i-1}$, то положить $\boldsymbol{i} = \boldsymbol{i}+1$ и перейти к шагу (2). В противном случае положить $\boldsymbol{N}_\varepsilon = \boldsymbol{N}_i$.

(4) Если $S \in \boldsymbol{N}_\varepsilon$, то выдать выход "ДА", иначе "НЕТ".

Так как $\boldsymbol{N}_\varepsilon \subseteq \boldsymbol{N}$ то алгоритм III.1.2.1 должен остановиться самое большее после $n+1$ повторений шага (2), если $\boldsymbol{N}$ содержит n нетерминалов.

Определение III.1.2.2. Символ $X \in \boldsymbol{N} \cup \boldsymbol{T}$ назовем *недостижимым* в БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$, если X не появляется ни водной выводимой цепочке.

Недостижимые символы можно устранить из БГ с помощью следующего алгоритма.

Алгоритм III.1.2.2. Устранение недостижимых символов.

Вход. БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$.

Выход. БГ $\boldsymbol{G}' = <\boldsymbol{N}', \boldsymbol{T}', \boldsymbol{P}', S>$,

(*i*) $\boldsymbol{L}(\boldsymbol{G}') = \boldsymbol{L}(\boldsymbol{G})$,

(*ii*) для всех $X \in \boldsymbol{N}' \cup \boldsymbol{T}'$ существуют такие цепочки α и β из $(\boldsymbol{N}' \cup \boldsymbol{T}')^*$, что $S \Rightarrow^*_{\boldsymbol{G}} \alpha X \beta$

Метод.

(1) Положить $V_0 = \{S\}$ и $i = 1$.

(2) Положить $V_i = \{X$: в P есть $A \to \alpha A \beta$ и $A \in V_{i-1}\} \cup V_{i-1}$.

(3) Если $V_i \neq V_{i-1}$, положить $i = i+1$ и перейти к шагу (2). В противном случае пусть $\boldsymbol{N}' = V_i \cap \boldsymbol{N}$, $\boldsymbol{T}' = V_i \cap \boldsymbol{T}$, $\boldsymbol{P}'$ состоит из правил множества правил $\boldsymbol{P}$, содержащих только символы из множества V_i, $\boldsymbol{G}' = <\boldsymbol{N}', \boldsymbol{T}', \boldsymbol{P}', S>$.

Теперь устраняются из БГ все бесполезные символы.

Алгоритм III.1.2.3. Устранение бесполезных символов.

Вход. БГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, у которой $\boldsymbol{L}(\boldsymbol{G}) \neq \emptyset$.

Выход. БГ $\boldsymbol{G}' = <\boldsymbol{N}', \boldsymbol{T}', P', S>$, у которой $\boldsymbol{L}(\boldsymbol{G}') = \boldsymbol{L}(\boldsymbol{G})$ и в $\boldsymbol{N}' \cup \boldsymbol{T}'$ нет бесполезных символов.

Метод.

(1) Применив к $\boldsymbol{G}$ алгоритм III.1.2.1, можно получить $\boldsymbol{N}_\varepsilon$. Положить $\boldsymbol{G} = <\boldsymbol{N} \cap \boldsymbol{N}_\varepsilon, \boldsymbol{T}, \boldsymbol{P}_1, S>$, где $\boldsymbol{P}_1$ состоит из правил множества $\boldsymbol{P}$, содержащих только символы из $\boldsymbol{N}_\varepsilon \cup \boldsymbol{T}$.

(2) Применив к G1 алгоритм III.1.2.2, получим $\boldsymbol{G}' = <\boldsymbol{N}', \boldsymbol{T}', P', S>$.

На шаге (1) алгоритма III.1.2.3 из $\boldsymbol{G}$ устраняются все нетерминалы, которые не могут порождать терминальных цепочек. Затем на шаге (2) устраняются все недостижимые символы. Каждый символ X результирующей грамматики должен появиться хотя бы в одном выводе вида $S \Rightarrow^* \gamma X \delta \Rightarrow^* \omega$. Заметим, что если сначала применить алгоритм III.1.2.2, а потом алгоритм III.1.2.1, то не всегда результатом будет грамматика, не содержащая бесполезных символов.

Примеры III.1.2.1. Рассмотрим БГ $\boldsymbol{G} = <\{S, A, B\}, \{a, b\}, \boldsymbol{P}, S>$, где $\boldsymbol{P} = \{S \to a$: $A, A \to AB, B \to b\}$.

Применим к $\boldsymbol{G}$ алгоритм III.1.2.3. На шаге (1) получим Ne=$\{S, B\}$ и $\boldsymbol{G}_1 = <\{S, B\}, \{a, b\}, \{S \to a, B \to b, S>$. Применив алгоритм III.1.2.2, получим $V_2 = V_1 = \{S, a\}$. Итак, $\boldsymbol{G}' = <\{S\}, \{a\}, \{S \to a\}, S>$.

Если применить к $\boldsymbol{G}$ сначала алгоритм III.1.2.2, то окажется, что все символы достижимы, так грамматика не изменится. Затем применение

алгоритма III.1.2.1 дает $\boldsymbol{N}_\varepsilon = \{S, B\}$ и результирующей будет граммматика $\boldsymbol{G}_1$,отличая от $\boldsymbol{G}'$.

Часто бывает удобно устранить из БГ $\boldsymbol{G}$ ε-правила, т.е. правила вида $A \to \varepsilon$. Но если $\varepsilon \in \boldsymbol{L}(\boldsymbol{G})$, то без правил вида $A \to \varepsilon$ не обойтись.

Определение III.1.2.3. Назовем БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$ *грамматикой без* ε*-правил* (или *неукорачивающей*), если либо

(1) $\boldsymbol{P}$ не содержит ε-правил, либо

(2) есть точно одно ε-правило $S \to \varepsilon$ и S не встречается в правых частях остальных правил из $\boldsymbol{P}$.

Алгоритм III.1.2.4. Преобразование в грамматику без ε-правил.

Вход. БГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$.

Выход. Эквивалентная БГ $\boldsymbol{G} = <\boldsymbol{N}', \boldsymbol{T}, \boldsymbol{P}', S'>$ без ε-правил.

Метод.

(1) Постройте $\boldsymbol{N}_\varepsilon = \{A: A \in \boldsymbol{N}\ A \Rightarrow^*_{\boldsymbol{G}} \varepsilon\}$. Это аналогично тому, что было в алгоритмах III.1.2.1 и III.1.2.2, и остается в качестве упражнения.

(2) Постройте $\boldsymbol{P}'$ так:

(а) Если $A \to \alpha_0 B_1 \alpha_1 B_2 \alpha_2 \ldots B_k \alpha_k$ принадлежит $\boldsymbol{P}$, $k \geq 0$ и $B_i \in \boldsymbol{N}_\varepsilon$ для $1 \leq i \leq k$, но ни один символ в цепочках α_j $(0 \leq j \leq k)$, не принадлежит $\boldsymbol{N}_\varepsilon$ то включить в $\boldsymbol{P}'$ все правила вида

$$A \to \alpha_0 X_1 \alpha_1 X_2 \ldots \alpha_{k-1} X_k \alpha_k,$$

где X_i – либо B, либо е, но не включать правило $A \to \varepsilon$ (это могло бы произойти в случае, если все α_i равны ε).

(б) Если $S \in \boldsymbol{N}_\varepsilon$, включить в $\boldsymbol{P}'$ правила $S' \to \varepsilon \mid S$, где S'– новый символ, и положить $\boldsymbol{N}' = \boldsymbol{N} \cup \{S'\}$. В противном случае положить $\boldsymbol{N}' = \boldsymbol{N}$ и $S' = S$.

(3) Положить $\boldsymbol{G}' = <\boldsymbol{N}', \boldsymbol{T}, \boldsymbol{P}', S'>$.

Примеры III.1.2.2. Рассмотрим БГ с правилами

$$S \to aSbS \mid bSaS \mid \varepsilon$$

Применяя к этой грамматике алгоритм 2 10, получаем грамматику

$$S' \to S \mid \varepsilon,$$

$$S\to aSbS|bSaS|aSb|abS|ab|bSa|baS|ba$$

Теорема III.1.2.1. Алгоритм III.1.2.4 дает грамматику без ε-правил, эквивалентную входной грамматике.

Другое полезное преобразование грамматик − устранение правил вида $A\to B$, которые мы будем называть *цепными*.

Алгоритм III.1.2.5. Устранение цепных правил.

Вход. БГ $\boldsymbol{G}$ без ε-правил.

Выход. Эквивалентная БГ $\boldsymbol{G'}$ без ε-правил и без цепных правил.

Метод.

(1) Для каждого $A\in\boldsymbol{N}$ можно построить $\boldsymbol{N}_A = \{B:\ A\Rightarrow^* B\}$ следующим образом:

(а) Положить $\boldsymbol{N}_0 =\{A\}$ и $i= 1$.

(б) Положить $\boldsymbol{N}_i= \{C:\ B\to C$ принадлежит $\boldsymbol{P}$ и $B\in\boldsymbol{N}_{i-1}\}\cup\boldsymbol{N}_{i-1}$.

(в) Если $\boldsymbol{N}_i \neq \boldsymbol{N}_{i-1}$, положить $i= i+1$ и повторить шаг (б). В противном случае положить $\boldsymbol{N}_A= \boldsymbol{N}_i$.

(2) Постройте $\boldsymbol{P'}$: если $B\to\alpha$ принадлежит $\boldsymbol{P}$ и не является цепным правилом, включить в $\boldsymbol{P'}$ правило $A\to\alpha$ для всех таких A, что $B\in\boldsymbol{N}_A$.

(3) Положить $\boldsymbol{G'}= <\boldsymbol{N'}, \boldsymbol{T}, \boldsymbol{P'}, S>$.

Примеры III.1.2.3. Применим алгоритм III.1.2.5 к грамматике $\boldsymbol{G}_0$ с правилами

$$E\to E+H|H,$$
$$H\to H*F|F,$$
$$F\to(E)|a.$$

На шаге (1) $\boldsymbol{N}_E = \{E, H, F\}$, $\boldsymbol{N}_T = \{ H, F\}$, $\boldsymbol{N}_F = \{F\}$. После шага (2) множество $\boldsymbol{P'}$ будет содержать:

$$E\to E+H|H*F|(E)|a,$$
$$H\to H*F\ |(E)|a,$$
$$F\to(E)|a.$$

Теорема III.1.2.2. Грамматика $\boldsymbol{G'}$, которую строит алгоритм III.1.2.5, не имеет цепных, правил и $\boldsymbol{L}(\boldsymbol{G'}) = \boldsymbol{L}(\boldsymbol{G})$.

Доказательство. Видно, что алгоритм III.1.2.5 дает грамматику $\boldsymbol{G'}$ без цепных правил. Покажем, что $\boldsymbol{L}(\boldsymbol{G'})\subseteq \boldsymbol{L}(\boldsymbol{G})$. Пусть $\omega\in\boldsymbol{L}(\boldsymbol{G'})$. Тогда в $\boldsymbol{G'}$ существует вывод $S \Rightarrow\alpha_0\Rightarrow\alpha_1\Rightarrow...\Rightarrow\alpha_N=\omega$. Если при переходе от α_i к α_{i+1} применяется правило $A\to\beta$, то существует такой символ $B\in\boldsymbol{N}$ (возможно, $B = A$), что $A\Rightarrow^*_{\boldsymbol{G}}B$ и $B\Rightarrow_{\boldsymbol{G}}\beta$. Таким образом, $A\Rightarrow^*_{\boldsymbol{G}}\beta$ и $\alpha_i\Rightarrow^*_{\boldsymbol{G}}\alpha_{i+1}$. Отсюда следует, что $S\Rightarrow^*_{\boldsymbol{G}}\omega$ и $\omega\in\boldsymbol{L}(\boldsymbol{G})$, так что $\boldsymbol{L}(\boldsymbol{G'})\subseteq\boldsymbol{L}(\boldsymbol{G})$.

Теперь покажем, что $\boldsymbol{L}(\boldsymbol{G})\subseteq\boldsymbol{L}(\boldsymbol{G'})$. Для этого возьмем $\omega\in\boldsymbol{L}(\boldsymbol{G})$ и $S\Rightarrow\alpha_0 \Rightarrow_{\boldsymbol{L}}\alpha_1 \Rightarrow_{\boldsymbol{L}}... \Rightarrow_{\boldsymbol{L}}\alpha_N= \omega$ – левый вывод цепочки ω грамматике $\boldsymbol{G}$. Можно найти последовательность индексов $i_1, i_2, ..., i_k$, состоящую в точности из тех j, для которых на шаге $\alpha_{j-1}\Rightarrow_l\alpha_j$, применяется не цепное правило. В частности, $i_k= \boldsymbol{N}$, так как вывод терминальной цепочки не может оканчиваться цепным правилом.

Так как мы рассматриваем левый вывод, то последовательные применения цепных правил заменяют символ, занимающий одну и ту же позицию в левовыводимых цепочках, из которых состоит соответствующая часть вывода в грамматике. Отсюда видно, что $S\Rightarrow_{\boldsymbol{G}'}\alpha_{i1}\Rightarrow_{\boldsymbol{G}'}\alpha_{i2}\Rightarrow_{\boldsymbol{G}'}...\Rightarrow_{\boldsymbol{G}'}\alpha_{ik}=\omega$. Итак, $\omega\in\boldsymbol{L}(\boldsymbol{G'})$ и, значит, $\boldsymbol{L}(\boldsymbol{G'})= \boldsymbol{L}(\boldsymbol{G})$.

Определения III.1.2.4. Пусть задана БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$. Тогда:

1. Нетерминал $A\in\boldsymbol{N}$ в $\boldsymbol{G}$ называется *циклическим символом*, если для него существует вывод $A\Rightarrow\zeta A\xi$, $\zeta\in(\boldsymbol{T}\cup\boldsymbol{N})^*$, $\xi\in(\boldsymbol{T}\cup\boldsymbol{N})^*$.
2. Циклический символ называется *эффективным*, если $A\Rightarrow\alpha A\beta$, где $|\alpha A\beta|>1$, иначе циклический символ называется фиктивным.
3. Грамматика $\boldsymbol{G}$ называется *циклической грамматикой*, если в ней имеется хотя бы один циклический символ.
4. Грамматика $\boldsymbol{G}$ называется *грамматикой без циклов*, если для нетерминала $A\in\boldsymbol{N}$ нет выводов вида $A\Rightarrow^+A$;
5. Грамматика $\boldsymbol{G}$ называется *приведенной грамматикой*, если она без циклов, без ε-правил и без бесполезных символов.

Грамматики с ε-правилами или циклами иногда труднее анализировать, чем грамматики без ε-правил, так как в любой практической ситуации бесполезные символы без необходимости увеличивают объем анализатора. Поэтому для некоторых алгоритмов

синтаксического анализа мы будем требовать, чтобы грамматики, фигурирующие в них, были приведенными. Докажем, что это требование все же позволяет рассматривать все КС-языки без циклов.

Теорема III.1.2.3. Если $\boldsymbol{L}-$ КС-язык, то $\boldsymbol{L}=\boldsymbol{L}(\boldsymbol{G})$ для некоторой приведенной БГ $\boldsymbol{G}$.

Доказательство. Применить к БГ, определяющей язык $\boldsymbol{L}$, алгоритмы III.1.2.2 − III.1.2.5.

Определение III.1.2.5. А-правилом КС–грамматики называется правило вида $A\to\alpha$ (не путайте A-правило с ε-правилом, которое имеет вид $B\to\varepsilon$).

Введем еще преобразование, с помощью которого можно удалить из грамматики одно правило вида $A\to\alpha B\beta$. Чтобы устранить это правило, надо добавить к грамматике новые правила, получающиеся из него заменой нетерминала B правыми частями всех B-правил.

Лемма III.1.2.1. Пусть $\boldsymbol{G}$ = <$\boldsymbol{N}$, $\boldsymbol{T}$, $\boldsymbol{P}$, S> – КС-грамматика и $\boldsymbol{P}$ содержит правило $A\to\alpha B\beta$, где $B\in\boldsymbol{N}$, а α и β принадлежат $(\boldsymbol{N}\cup\boldsymbol{T})^*$. Пусть $B\to\gamma_1|\gamma_2|\ \dots\ |\gamma_k|$ – все B-правила этой грамматики. Пусть $\boldsymbol{G}'$= <$\boldsymbol{N}$, $\boldsymbol{T}$, $\boldsymbol{P}$, S>, где
$\boldsymbol{P}'=(\boldsymbol{P}\backslash\{A\to\alpha B\beta\})\cup\{A\to\alpha\gamma_1\beta|\alpha\gamma_2\beta|\ \dots\ |\alpha\gamma_k\beta|\}$. Тогда $\boldsymbol{L}(\boldsymbol{G})=\boldsymbol{L}(\boldsymbol{G}')$.

Доказательство этого утверждения проводится самостоятельно в качестве задания.

Примеры III.1.2.4. Устраним правило $A\to aAA$ из грамматики $\boldsymbol{G}$, имеющей два правила $A\to aAA|b$. Применим лемму, полагая $\alpha=a$ и $\beta=A$, получим грамматику $\boldsymbol{G}'$ с правилами $A\to aAAA|abA|b$.

Задания III.1.2:

1. Постройте приведенную грамматику, эквивалентную грамматике, имеющей следующие правила:

$S\to aABS|bCACd,$
$A\to bAB|cSA|cCC,$
$B\to bAB|cSB,$

$C \to cS \mid c.$

2. Постройте приведенную грамматику, эквивалентную грамматике, имеющей следующие правила:

$S \to aAB \mid E,\ A \to dDA \mid \varepsilon,$
$B \to bE \mid f,$
$C \to cAB \mid dSD \mid a,$
$D \to \varepsilon A,$
$E \to fA \mid g.$

3. Доказать, что язык, порождаемый циклической приведенной КС-грамматикой, содержащей хотя бы один эффективный циклический символ, бесконечен.

4. Доказать, что не циклическая БГ порождает конечный язык.

5. Постройте приведенную грамматику, порождающую идентификаторы, состоящие из букв цифр.

Вопросы III.1.2:

1. Порождает ли не циклическая БГ конечный язык?

2. Порождает ли циклическая приведенная БГ, содержащая хотя бы один эффективный циклический символ, бесконечный язык?

3. Можно ли преобразовать правила

$A \to AA \mid \alpha,\ A \to A\alpha A \mid \beta,\ A \to \alpha A \mid A\beta \mid \gamma$

так, чтобы получить неоднозначную грамматику?

4. Можно ли построить циклическую приведенную грамматику для порождения языка

$\boldsymbol{L} = \{a^{2n}\, b^m\, c^{2k}:\ m = n + k,\ m > 1\}$?

5. Будет ли грамматика с правилами неоднозначной

$S \to abC \mid aB,\ B \to bc,\ C \to bc$?

6. Является ли контекстно-свободным язык

$\{\omega\omega\omega:\ \omega \in \{a, b\}^*\}$?

7. Является ли контекстно-свободным язык

$\{\gamma\delta\delta\omega:\ \gamma \in \{a, b\}^+,\ \delta \in \{a, b\}^+,\ \omega \in \{a, b\}^+\}$?

8. Существуют ли такие бесконтекстные языки $\boldsymbol{L}_1 \subseteq \{a, b\}^*$ и $\boldsymbol{L}_2 \subseteq \{b, c\}^*$, что язык $\boldsymbol{L}_1 - \boldsymbol{L}_2$ не является бесконтекстным?

9. Существует ли над алфавитом $\{a, b\}$ такой бесконтекстный язык $\boldsymbol{L}$, что язык $\{\gamma b\delta: \gamma \in \boldsymbol{L}, \gamma\delta \in \boldsymbol{L}\}$ не является бесконтекстным?

III.1.3. Нормальная форма Хомского

В этом параграфе будет определена нормальная форма Хомского бесконтекстной грамматики, описывается алгоритм преобразования к этой форме, а также будут предложены примеры, даны задания, сформулированы вопросы [1-9,11-13,15-22,24,25, 27-32].

Определение III.1.3.1. БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$ называется грамматикой в *нормальной форме Хомского* (или в *бинарной нормальной форме*), если каждое правило из $\boldsymbol{P}$ имеет один из следующих видов:

(1) $A \to BC$, где A, B и C принадлежат $\boldsymbol{N}$,

(2) $A \to a$, где $a \in \boldsymbol{T}$,

(3) $S \to \varepsilon$, если $\varepsilon \in \boldsymbol{L}(\boldsymbol{G})$, S нет в правых частях правил.

Покажем, что каждый КС-язык порождается грамматикой в нормальной форме Хомского.

Алгоритм III.1.3.1. Преобразование к нормальной форме Хомского.

Вход. Приведенная БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$.

Выход. БГ $\boldsymbol{G}'$ в нормальной форме Хомского, эквивалентная $\boldsymbol{G}$, т.е. $\boldsymbol{L}(\boldsymbol{G}') = \boldsymbol{L}(\boldsymbol{G})$.

Метод. Грамматика $\boldsymbol{G}'$ строится по $\boldsymbol{G}$ следующим образом:

(1) Включить в $\boldsymbol{P}'$ каждое правило из $\boldsymbol{P}$ вида $A \to a$.

(2) Включить в $\boldsymbol{P}'$ каждое правило из $\boldsymbol{P}$ вида $A \to BC$.

(3) Включить в $\boldsymbol{P}'$ правило $S \to \varepsilon$, если оно было в $\boldsymbol{P}$.

(4) Для каждого правила из $\boldsymbol{P}$ вида $A \to X_1 \ldots X_k$, где $k>2$, включить в $\boldsymbol{P}'$ следующие правила:

$$A \to X'_1 <X_2 \ldots X_k>$$
$$<X_2 \ldots X_k> \to X'_2 <X_3 \ldots X_k>$$
$$\ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$$
$$<X_{k-2}X_{k-1} \ldots X_k> \to X'_{k-2} <X_{k-1}X_k>$$
$$<X_{k-1}X_k> \to X'_{k-1}X'_k$$

где $X'_i = X_i$, если $X_i \in \boldsymbol{N}$; X'_i – новый нетерминал, если $X_i \in \boldsymbol{T}$; $<X_i \ldots X_k>$ – новый нетерминал.

(5) Для каждого правила вида $A \to x_1 x_2$, где хотя бы один из символов $x_1 \in \boldsymbol{T}$ и $x_2 \in \boldsymbol{T}$, включить в $\boldsymbol{P}'$ правило $A \to x'_1 x'_2$.

Для каждого нетерминала вида a', введенного на шагах (4) и (5), включить в $\boldsymbol{P}'$ правило $a' \to a$. Пусть $\boldsymbol{N}'$ – это $\boldsymbol{N}$ вместе со всеми новыми нетерминалами, введенными при построении $\boldsymbol{P}'$. Тогда искомой грамматикой будет $\boldsymbol{G} = <\boldsymbol{N}', \boldsymbol{T}, \boldsymbol{P}', S>$.

Теорема III.1.3.1. Пусть $\boldsymbol{L}$− КС-язык. Тогда $\boldsymbol{L} = \boldsymbol{L}(\boldsymbol{G}')$ для некоторой БГ $\boldsymbol{G}'$ в нормальной форме Хомского.

Доказательство: По теореме *III*.1.2.3 $\boldsymbol{L}$определяется приведенной грамматикой $\boldsymbol{G}$. Алгоритм III.1.3.1 строит по ней грамматику $\boldsymbol{G}'$, имеющую нормальную форму Хомского. Остается показать, что $\boldsymbol{L}(\boldsymbol{G}) = \boldsymbol{L}(\boldsymbol{G}')$. Это доказывается применением леммы *III*.1.2.1 к каждому правилу грамматики $\boldsymbol{G}'$, в правую часть которого входит a', а затем к правилам с нетерминалами вида $<X_i \dots X_j>$.

Примеры III.1.3.1. Приведенная БГ $\boldsymbol{G}$ имеет следующие правила $S \to aAB|BA$, $A \to BBB|a$, $B \to AS|b$. Строится $\boldsymbol{P}'$ алгоритм III.1.3.1, сохраняя правила $S \to BA$, $A \to a$, $B \to AS$ и $B \to b$. Заменяем $S \to aAB$ правилами $S \to a'<AB>$ и $<AB> \to AB$, а $A \to BBB$ -правилами $A \to B<BB>$ и $<BB> \to BB$. Наконец, добавляем $a' \to a$. В результате получаем грамматику $\boldsymbol{G} = <\boldsymbol{N}', \{a, b\}, \boldsymbol{P}', S>$, где $\boldsymbol{N}' = \{S, A, B, <AB>, <BB>, a'\}$, а $\boldsymbol{P}'$ состоит из правил $S \to a'<AB>| BA$, $A \to B<BB>|a$, $B \to AS|b$, $<AB> \to AB$, $<BB> \to BB$, $a' \to a$

Задания III.1.3. Преобразовать в нормальную форму Хомского БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$ со следующими параметрами:

1. $\boldsymbol{T} = \{a, b\}$, $\boldsymbol{N} = \{S\}$, $\boldsymbol{P} = \{S \to ab, S \to aSb, S \to SS\}$;
2. $\boldsymbol{T} = \{x, a, p\}$, $\boldsymbol{N} = \{S, D\}$, $\boldsymbol{P}, \{A\}>$, $\boldsymbol{P} = \{S \to x|Dx, D \to Da|p\}$;
3. $\boldsymbol{T} = \{a, b\}$, $\boldsymbol{N} = \{A, B, S\}$, $\boldsymbol{P} = \{S \to Aa|a|b, A \to Ab|a\ |aB, B \to B\text{a}|b\}$.

Вопросы III.1.3:

1. Существует ли нормальны форма Хомского для любой БГ?
2. Как строится приведенная форма БГ?
3. Что является входом для алгоритма III.1.3.1?

III.1.4. Нормальная форма Грейбах

В этом параграфе будет определена нормальная форма Грейбах бесконтекстной грамматики, описывается алгоритм преобразования к нормальной формы Грейбах, а также будут предложены примеры, даны задания, сформулированы вопросы [1-9,11-13,15-22,24,25].

Определение III.1.4.1. Нетерминал A БГ $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$ называется *рекурсивным*, если $A \Rightarrow^{*} \alpha A \beta$ для некоторых α и β. Если $\alpha=\varepsilon$, то A называется *леворекурсивным*. Аналогично, если $\beta=\varepsilon$, то A называется *праворекурсивным*. Грамматика, имеющая хотя бы один леворекурсивный нетерминал, называется *леворекурсивной*. Аналогично определяется *праворекурсивная* грамматика. Грамматика, в которой все нетерминалы, кроме, быть может, начального символа, рекурсивные, называется *рекурсивной*.

Некоторые из обсуждаемых далее алгоритмов разбора не могут работать с леворекурсивными грамматиками. Покажем, что каждый КС-язык определяется хотя бы одной не леворекурсивной грамматикой. Начнем с устранения в БГ непосредственной леворекурсивности.

Примеры III.1.4.1. Пусть задана грамматика с правилами

$$E \to E+H|H,\ H \to H * F|F,\ F \to (E)|a.$$

Если применить к ней конструкцию леммы III.1.2.9, то получится эквивалентная ей грамматика $\boldsymbol{G'}$ с правилами

$$E \to H|HE',\ E' \to +H|+HE',\ H \to F|FH',\ H' \to * F|FH',\ F \to (E)|a.$$

Алгоритм III.1.4.1. Устранение левой рекурсии.

Вход. Приведенная БГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$.

Выход. Эквивалентная БГ без левой рекурсии.

Метод. Имеется пять видов входа:

(1) Пусть $\boldsymbol{N} = \{A_1, ..., A_N\}$. Преобразуем $\boldsymbol{G}$ так, чтобы в правиле $A_i \to \alpha$ цепочка α начиналась либо с терминала, либо с такого A_j, что $j>i$. С этой целью положим i=1.

(2) Пусть множество A_i – правил – это $A_i \to A_i\alpha_1|...|\ A_i\alpha_m|\beta_1|...|\ \beta_P$, где ни одна из цепочек β_j не начинается с A_k, если $k \le i$. (Это всегда можно сделать.) Заменим A_i – правила правилами

$$A_i \to \beta_1|...|\ \beta_P|\beta_1 A'_i|...|\ \beta_P A'_i,\ \ A'_i \to \alpha_1|...|\ \alpha_m|\alpha_1 A'_i|...|\ \alpha_m A'_i$$

где A'_i – новый нетерминал. Правые части всех A_i – правил начинаются теперь с терминала или с A_k для некоторого $k>i$.

(3) Если $i=N$, полученную грамматику G' считать результатом и остановиться. В противном случае положить $i = i+1$ и $j = 1$.

(4) Заменить каждое правило вида $A_i \to A_j\alpha$ правилами, где $A_i \to \beta_1\ \alpha|\ ...|\ \beta_m\ \alpha$, где $A_j \to \beta_1|...|\ \beta_m$ – все A_j – правила. Так как правая часть каждого правила A_j начинается уже с терминала или с A_k для $k>j$, то его правая часть будет теперь обладать этим свойством.

(5) Если $j=i-1$, перейти к шагу (2). В противном случае положить $j=j+1$ и перейти к шагу (4).

Теорема III.1.4.1. Каждый КС-язык порождается не леворекурсивной грамматикой.

Примеры III.1.4.2. Пусть бесконтекстная грамматика определяется правилами

$$A \to BC|a$$
$$B \to CA|Ab$$
$$C \to AB|CC|a$$

Положим $A_1 = A$, $A_2 = B$ и $A_3 = C$. После каждого применения шага (2) или (4) алгоритма III.1.4.1 получаются такие грамматики:

Шаг (2) для $i = 1$: G не меняется

Шаг (4) для $i = 2$, $j = 1$: $B \to CA|BCb|ab$

Шаг (2) для $i = 2$: $B \to CA|ab|CAB'|abB'$, $B' \to CbB'|Cb$

Шаг (4) для $i = 3$, $j = 1$: $C \to BCB|aB|CC|a$

Шаг (4) для $i = 3$, $j = 2$: $C \to CACB|abCB|CAB'CB\ |abB'CB\ |\ aB|CC|a$

Шаг (2) для $i = 3$:

$C \to abCB|abB'CB\ |aB\ |a|abCBC'|abB'CBC'|aBC'|aC'$

$C' \to ACBC'|AB'CBC'|CC'|ACB|AB'CB|C$

Интересный частный случай не леворекурсивной грамматики – грамматика в нормальной форме Грейбах.

Определение III.1.4.2. *БГ* $G = <T, N, P, S>$ называется грамматикой в *нормальной форме Грейбах*, если в ней нет ε-правил и каждое правило из P, отличное от $S \to \varepsilon$, имеет вид $A \to a\alpha$, где $a \in T$ и $\alpha \in N^*$.

Если грамматика не леворекурсивна, то на множестве ее нетерминалов можно определить естественный частичный порядок. Этот частичный порядок можно вложить в линейный порядок, полезный при преобразовании грамматики к нормальной форме Грейбах.

Лемма III.1.4.1. Пусть $G = <T, N, P, S>$ – не леворекурсивная грамматика. Существует такой линейный порядок $<$ на N, что если $A \to B\alpha$ принадлежит P, то $A < B$.

Доказательство. Пусть R – такое отношение на N, что ARB тогда и только тогда, когда $A \Rightarrow^{+} B\alpha$ для некоторого α. Так как грамматика G не леворекурсивна, то R – частичный порядок. Отношение R можно расширить до линейного порядка $<$, обладающего нужным свойством.

Алгоритм III.1.4.2. Преобразование к нормальной форме Грейбах.

Вход. Не леворекурсивная приведенная БГ $G = <N, T, P, S>$.

Выход. Эквивалентная БГ G' в нормальной форме Грейбах.

Метод. Имеются шесть видов входа:

(1) Постройте с помощью леммы III.1.4.1 такой линейный порядок $<$ на N, что каждое A-правило начинается либо с терминала, либо с такого нетерминала B, что $A < B$. Упорядочить $N = \{A_1, ..., A_N\}$ так, что $A_1 < A_2 < ... < A_N$.

(2) Положить $i = N - 1$.

(3) Если $i = 0$, перейти к шагу (5). В противном случае заменить каждое правило вида $A_i \to A_j\alpha$, где $j > i$, правилами $A_i \to \beta_1\alpha | ... | \beta_m\alpha$, где

$A_j \to \beta_1 | ... | \beta_m$ – все A_j – правила. Позже мы убедимся, что каждая из β_1, ..., β_m цепочек начинается терминалом.

(4) Положить $i = i$-1 и вернуться к шагу (3).

(5) Сейчас правая часть каждого правила (кроме, возможно, $S \to \varepsilon$) начинается терминалом. В каждом правиле $A \to aX_1 \ldots X_k$ заменить $X_j \in \boldsymbol{T}$ новым нетерминалом X'_j.

Для новых нетерминалов X'_j, введенных на шаге (5), добавить правила $X'_j \to X_j$.

Теорема III.1.4.1. Если $\boldsymbol{L}$ – КС-язык, то $\boldsymbol{L} = \boldsymbol{L}(\boldsymbol{G})$ для некоторой грамматики $\boldsymbol{G}$ в нормальной форме Грейбах.

Доказательство. Индукцией по $\boldsymbol{N} - i$ (т. е. по $\boldsymbol{i}$, но в обратном порядке, начиная с $i = \boldsymbol{N}$-1 и кончая i=1) можно показать, что после выполнения шага (3) алгоритма 2.14 для i правая часть каждого A_i-правила начинается терминалом. Ключевой момент здесь – использование линейного порядка <. После шага (5) грамматика преобразуется к нормальной форме Грейбах и по лемме 2.14 порождаемый ею язык не изменится.

Примеры III.1.4.3. Рассмотрим грамматику $\boldsymbol{G}$ с правилами

$$E \to H | HE'$$
$$E' \to +H | +HE'$$
$$H \to F | FH'$$
$$H' \to *F | *FH'$$
$$F \to (E) | a$$

Упорядочим нетерминалы следующим образом:

$$E' < E < H' < H < F.$$

Правая часть каждого F – правила начинается терминалом, как и должно быть, так как F – наибольший нетерминал в этом упорядочении. Предшествующий ему нетерминал $\boldsymbol{T}$ имеет правила $H \to F | FH'$, так что, заменив в них F, получаем $H \to (E) | a(E)H' | aH'$. Переходя к E', обнаруживаем, что здесь ничего менять не надо. Затем заменяем E – правила правилами

$$E \to (E) | a | (E)H' | aH' | (E)E' | aE' | (E)H'E' | aH'E'B\ E'$$

правилах ничего не менять не надо.

На шагах (5) и (6) появляются новый нетерминал)'и правило)' →), поэтому все вхождения) в предыдующих правилах надо заменить в нормальной форме Грейбах с правилами:

$E \rightarrow (E) | a | (E)H' | aH' | (E)E' | aE' | (E)H'E' | aH'E'$

$E' \rightarrow + H | + HE'$

$H \rightarrow (E)' | a | (E)'H' | aH'$

$H' \rightarrow *F | *FH'$

$F \rightarrow (E)' | a$

$)' \rightarrow)$

В дальнейшем нормальная форма Грейбах БГ в разделе III.3.1 будет использоваться для построения стекового автомата, эквивалентного заданной бесконтекстной грамматике, что существенно облегчит разработку языковых процессоров.

Задания III.1.4:

1. Преобразовать в нормальную форму Грейбах КС-грамматику $\boldsymbol{G}$ = <{*a,b,c*}, {*A, B, C, S*}, {*S→aAB,A→aA|bB,*B*→ACb|b,C→bA|Cc*}, *S*>.

2. Построить эквивалентную для леворекурсивной грамматики с правилами *S→SabA|ba, A→AbA|bAa|c* праворекурсивную грамматику.

3.Постройте алгоритм устранения правой рекурсии и доказать эквивалентность этого преобразования.

Вопросы III.1.4:

1. Существует ли БГ, в которой были бы выводимы цепочки вида 00..0011..1122..22, в которых числа 0, 1 и 2 равны, и только они?

2. Является ли БГ лево(право)рекурсивной с правилами S→*Aab|bSb|abB, A→ab*S|*ba|bbA, B→bB|Da|Aa, D→Dbb|aDa*

3. Есть ли КС-грамматику для порождения языка $\{a^n b^n c^n | n>1\}$?

4. Как определяется нормальная форма Грейбах в контекстно-свободной грамматике?

5. Что является входом алгоритма преобразования к нормальной форме Грейбах?

III.2. Распознающие механизмы бесконтекстных языков

III.2.1. Состав и структура стековых автоматов

В этом параграфе будут описаны состав, структура, функция перехода, конфигурация и такт стекового автомата, предложены примеры, даны задания, сформулированы вопросы [1-13,15-21,25,31].

Стековый автомат (СА) – это односторонний распознаватель, распознающий контекстных языков. Его устройство близко к устройству конечного автомата, но он еще снабжен дополнительной рабочей ленты – стековой памятью которая организована по принципу "*последним войти, первым выйти – last input, first output* (*LIFO*)". Доступ к стековой памяти осуществляется только через ее верхней ячейки, которая называется *вершиной стека*, т.е. символы вводится (заталкивается) только на вершину и выводится (выталкивается) только с вершины стековой памяти. Для этого в стековом автомате предусмотрена дополнительная головка, всегда указывающая на вершину стека.

Данные в стеке можно представить в виде конечной цепочки символов h_i ($0 \le i \le n$) в конечном алфавите $\boldsymbol{H}$ и считается, что на вершине стека будет находиться самый левый символ цепочки. Причем, если i-й символ от последнего, записанного в стеке, должен быть прочитан, то i-1 промежуточных символов должны сначала быть стерты. После вывода последнего символа из стека внутри нее останется пустая цепочка, которую показывает маркер дна стека.

Итак, в составе стекового автомата имеются *конечная входная лента*, *внутренняя память*, *внешняя память*, *правосторонная головка для входной ленты*, *управляющее устройство*, *стековая память* и *головка для стека*. Состав и структура стекового автомата показана на рисунке III.2.1.

Стековый автомат начинает свою работу в начальном состоянии, имея цепочку входных символов ленте и пустой стек. Последующие конфигурации определяются прочитанным входным символом, символом внутреннего состояния управляющего устройства и символом

на вершине стека. За один такт работы автомата головка стека может произвести следующие действия:

1) стереть символ из вершины стека, при этом все оставшиеся в стеке символы перемещаются на одну ячейку вверх;

2) стереть символ из вершины и записать на рабочую ленту непустую цепочку символов, при этом содержимое стека сдвигается вниз ровно настолько, какова длина записываемой цепочки.

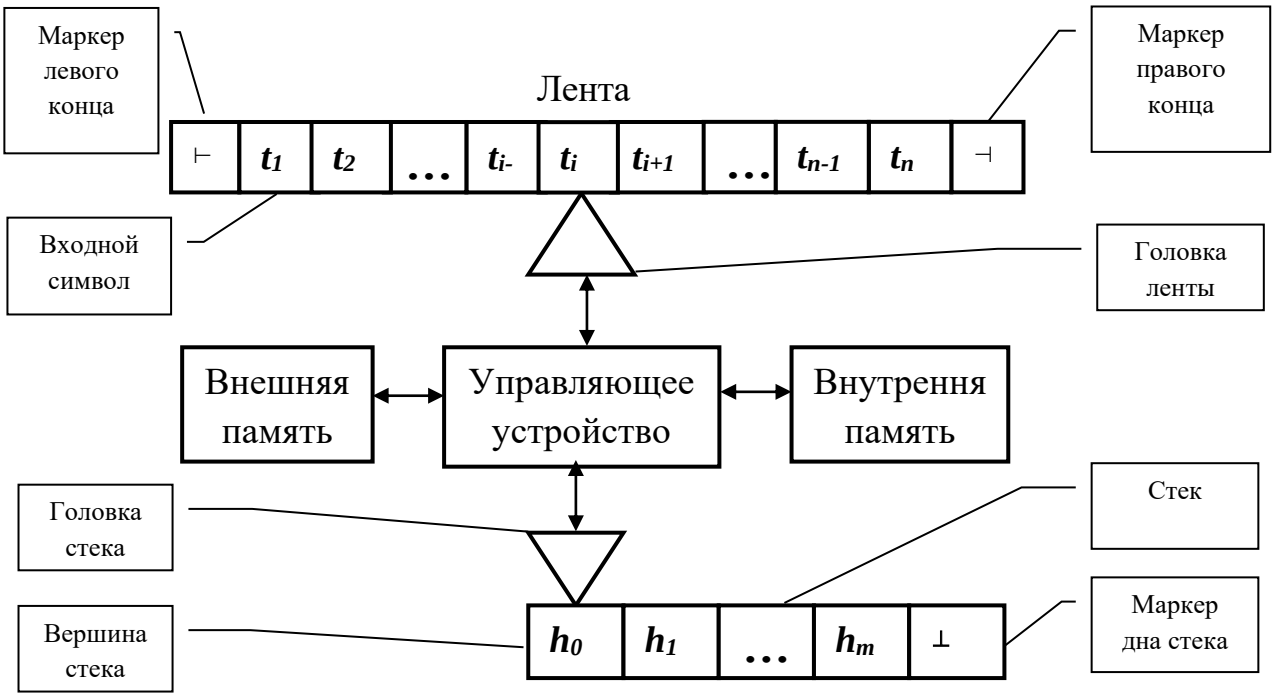


Рисунок III.2.1. Состав и структура стекового автомата

Определения III.2.1.1.

1.Стековый автомат в терминах функции перехода определяется десяткой $\boldsymbol{M} = \langle \boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \delta, \vdash, \dashv, \perp \rangle$,

2. Стековый автомат в терминах отношении перехода определяется десяткой $\boldsymbol{M} = \langle \boldsymbol{Q}, \boldsymbol{T}, I, \boldsymbol{F}, \boldsymbol{H}, h_0, \boldsymbol{\Delta}, \vdash, \dashv, \perp, \delta \rangle$, где:

$\boldsymbol{Q}$ – конечное множество состояний управляющего устройства;

$\boldsymbol{T}$ – конечное множество входных символов, $\boldsymbol{Q} \cap \boldsymbol{T} = \emptyset$, $\perp \notin \boldsymbol{T}$;

q_s – символ начального состояния управляющего устройства, $q_s \in \boldsymbol{Q}$;

I – множество начальных состояний управляющего устройства, $\boldsymbol{I} \subseteq \boldsymbol{Q}$;

$\boldsymbol{F}$ – множество финальных состояний управляющего устройства, $\boldsymbol{F} \subseteq \boldsymbol{Q}$;

q_f – символ финальльного состояния управляющего устройства, $q_f \in \boldsymbol{F}$;

$\boldsymbol{H}$ – конечное множество стековых символов;

h_0 – начальный символ на вершине стека, $h_0 \in \boldsymbol{H}$;

$\vdash, \dashv$ – маркер начало, маркер конца ленты, $\vdash, \dashv \notin \boldsymbol{T}$;

ε – пустая цепочка;

$\perp$ – маркер дна стека;

δ – функция перехода недетерминированного стекового автомата (НСА) отображает множество $\boldsymbol{Q}\times(\boldsymbol{T}\cup\{\varepsilon\})\times\boldsymbol{H}$ в множество всех конечных подмножеств множества $\boldsymbol{Q}\times\boldsymbol{H}^*$, т.е.

$$\delta: \boldsymbol{Q}\times(\boldsymbol{T}\cup\{\varepsilon\})\times\boldsymbol{H}\rightarrow \mathcal{P}(\boldsymbol{Q}\times\boldsymbol{H}^*),$$

где $\mathcal{P}(\boldsymbol{Q}\times\boldsymbol{H}^*)$ – множество всех конечных подмножеств множества $\boldsymbol{Q}\times\boldsymbol{H}^*$.

δ – функция перехода детерминированного стекового автомата (ДСА) отображает множество $\boldsymbol{Q}\times(\boldsymbol{T}\cup\{\varepsilon\})\times\boldsymbol{H}$ в множество $\boldsymbol{Q}\times\boldsymbol{H}^*$, т.е. δ: $\boldsymbol{Q}\times(\boldsymbol{T}\cup\{\varepsilon\})\times\boldsymbol{H}\rightarrow\boldsymbol{Q}\times\boldsymbol{H}^*$.

$\boldsymbol{\Delta} \subseteq (\boldsymbol{Q}\times\boldsymbol{T}^*\times\boldsymbol{H}^*) \times (\boldsymbol{Q}\times\boldsymbol{H}^*)$ – множество перехода.

Замечание III.2.1. Стековые автоматы можно изображать в виде диаграмм состояний. На диаграмме каждое состояние обозначается кружком, а переход - стрелкой. Каждое начальное состояние распознается по ведущей в него короткой стрелке. Каждое допускающее (финальное) состояние отмечается на диаграмме двойным кружком. Стрелка с пометкой x, δ: γ ведущая из p в q, показывает, что $\langle\langle p, x, \delta\rangle, \langle q, \gamma\rangle\rangle$ является переходом данного стекового автомата.

Примеры III.2.1. Пусть $\boldsymbol{Q}$ ={1, 2}, $\boldsymbol{T}$ ={a, b}, $\boldsymbol{H}$ ={A, B}, $\boldsymbol{I}$=1, $\boldsymbol{F}$=2, $\boldsymbol{\Delta}$={$\langle\langle 1, a, \varepsilon\rangle, \langle 1, A\rangle\rangle$, $\langle\langle 1, b, \varepsilon\rangle, \langle 1, B\rangle\rangle$, $\langle\langle 1, \varepsilon, \varepsilon\rangle, \langle 2, \varepsilon\rangle\rangle$, $\langle\langle 2, a, A\rangle, \langle 2, \varepsilon\rangle\rangle$,$\langle\langle 2, b, B\rangle, \langle 2, \varepsilon\rangle\rangle$}. Диаграмма стекового автомата показана на рисунке III.2.1.

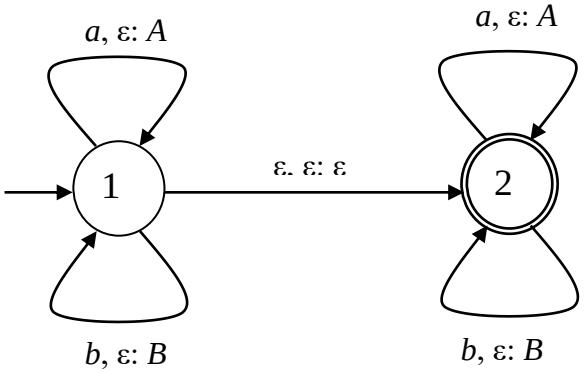


Рисунок II.2.1.10.Диаграмма стекового автомата

Пусть задан СА $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$, где $q \in \boldsymbol{Q}$, $t \in \boldsymbol{T}$, $\tau \in \boldsymbol{T}^*$, $h \in \boldsymbol{H}$, $\eta \in \boldsymbol{H}^*$, $q_i \in \boldsymbol{Q}$, $\eta_i \in \boldsymbol{H}^*$, $1 \leq i \leq m$. Тогда *такт* СА $\boldsymbol{M}$ в терминах функции перехода выражается так:

1) запись $\delta(q, t, h) = \{(q_1, \eta_1), (q_2, \eta_2), \ldots, (q_m, \eta_m)\}$ означает, что СА, находясь в состоянии q, считывает символ на вершине стека h и на входной ленте символ t, переходит в состояние q_i, заменяя в стеке символ h на цепочку η_i и продвигает головку на одну ячейку вправо;

2) запись $\delta(q, \varepsilon, h) = \{(q_1,\eta_1), (q_2,\eta_2), \ldots, (q_m,\eta_m)\}$ означает, что СА в состоянии q при символе на вершине стека h независимо от считываемого входного символа перейдет в состояние q_i, заменит в стеке символ h на η_i, а головка не перемещается. Такая операция используется для изменения содержания стека и называется ε-*движением*.

3) если стек пуст, т.е. $\eta_i = \varepsilon$, то следующий такт не возможен.

Если символ h заменено на цепочку η_i, то ее самый левый символ окажется на вершине стека, а самый правый – на дне стека.

Определение III.2.1.2. Если $q \in \boldsymbol{Q}$ – текущее состояние управляющего устройства, $q_s \in \boldsymbol{Q}$ – начальное состояние управляющего устройства, $q_f \in \boldsymbol{F}$ – финальное состояние управляющего устройства, $\tau \in \boldsymbol{T}^*$ – неиспользованная цепочка на входной ленте (ее самый левый символ обозревается головкой, если $\tau = \varepsilon$, то считается, что входная цепочка полностью прочитана), $\eta \in \boldsymbol{H}^*$ – цепочка в стеке (считается, что ее самый левый символ будет символом на вершине стека, если $\eta = \varepsilon$, то считается стек пуст) и $h_0 \in \boldsymbol{H}$ – символ на вершине стека, то следующая тройка:

$(q_s, \tau, h_0) \in \boldsymbol{Q} \times \boldsymbol{T}^* \times \boldsymbol{H}^*$ – *начальная конфигурация* стекового автомата;

$(q, \tau, \eta) \in \boldsymbol{Q} \times \boldsymbol{T}^* \times \boldsymbol{H}^*$ – *текущая конфигурация* стекового автомата;

$(q_f, \varepsilon, \eta) \in \boldsymbol{Q} \times \boldsymbol{T}^* \times \boldsymbol{H}^*$ – *заключительная конфигурация* стекового автомата.

В начальной конфигурации СА находится в начальном состоянии содержит на ленте цепочку входных символов и пустой стек. Последующие конфигурации автомата определяются прочитанным символом состояния управляющего устройства, входным символом на ленте и символом на вершине стека.

За один такт работы автомата головка стека может произвести следующие действия:

1) стереть символ из вершины, при этом все символы, оставшиеся в стеке, перемещаются на одну ячейку вверх;

2) стереть символ из вершины и записать в стек непустую цепочку символов, при этом содержимое стека сдвигается вниз ровно настолько, какова длина с записываемой цепочки.

Такт стекового автомата $\boldsymbol{A}$ можно выразить в терминах бинарного отношения $\models_{\boldsymbol{A}}$, определенного над конфигурациями из множества $\boldsymbol{Q}\times\boldsymbol{T}^*\times\boldsymbol{H}^*$, здесь, если заранее известен стековый автомат $\boldsymbol{A}$, то $\boldsymbol{A}$ при $\models$ можно опустить.

Пусть задан стековый автомат $\boldsymbol{A}$, в котором $q\in\boldsymbol{Q}$, $q'\in\boldsymbol{Q}$, $t\in\boldsymbol{T}\cup\{\varepsilon\}$, $\tau\in\boldsymbol{T}^*$, $h\in\boldsymbol{H}$, $\eta\in\boldsymbol{H}^*$, тогда:

1) если $\tau\neq\varepsilon$, $\tau=t\tau'$, $\tau'\in\boldsymbol{T}^*$, $\eta\neq\varepsilon$, $\eta=h\eta'$, $\eta'\in\boldsymbol{H}^*$ и $\xi\in\boldsymbol{H}^*$, то его такт записывается как отношение $(q,\ t\tau',\ h\eta') \models (q',\ \tau',\ \xi\eta')$, которое означает, что стековый автомат, находясь в состоянии q, обозревая входной символ t и считывая на вершине стека символ h, совершает такт в котором переходит в состояние q', продвигает головку на одну ячейку вправо и заменяет на вершине стека символ h на цепочку ξ;

2) если $\tau=\varepsilon$, $\eta\neq\varepsilon$, $\eta=h\eta'$, $\eta'\in\boldsymbol{H}^*$ и $\xi\in\boldsymbol{H}^*$, то его такт записывается как отношение $(q,\ \varepsilon,\ h\eta') \models (q',\ \varepsilon,\ \xi\eta')$, которое означает, что входная цепочка полностью прочитана и стековый автомат совершает *ε-такт*, в котором не сдвигает головку, но изменяет состояние управляющего устройства q на состояние q' и символ на вершине стека h заменяет на цепочку ξ.

3) если $\eta=\varepsilon$, т.е. если стек пуст, то следующий такт не возможен.

Можно определить k-тую степень бинарного отношения $\models$, обозначив ее как $\models^k$: если для заданной входной цепочки $t_1t_2 \ldots t_k\tau$ найдутся последовательность состояний $q_1, q_2, \ldots, q_k, q_{k+1}$ из множества $\boldsymbol{Q}$ и последовательность цепочек $\eta_1, \eta_2 \ldots, \eta_k, \eta_{k+1}$ из множества $\boldsymbol{H}^*$ такие, что $(q_1, t_1 t_2 \ldots t_k\tau, \eta_1) \models (q_2, t_2 \ldots t_k\tau, \eta_2) \models \ldots \models (q_k, t_k\tau, \eta_k) \models q_{k+1}, \tau, \eta_{k+1})$, то можно написать $(q_1, t_1t_2 \ldots t_k\tau, \eta_1) \models^k (q_{k+1}, \tau, \eta_{k+1})$.

Далее, если для любых $i \geq 1$ или $i \geq 0$ выполняется бинарное отношение $(q_0, \tau, h_0) \models^i (q_i, \varepsilon, \eta)$, то его можно записать в виде $(q_0, \tau, h_0) \models^+ (q_i, \varepsilon, \eta)$ или $(q_0, \tau, h_0) \models^* (q_i, \varepsilon, \eta)$ соответственно, здесь $q_0 = q_s$, $q_i \in \boldsymbol{F}$, $\tau \in \boldsymbol{T}^*$, $h_0 \in \boldsymbol{H}$, $\eta \in \boldsymbol{H}^*$ и для отношения $\models$ обозначение $\models^+$ – транзитивное замыкание, а $\models^*$ – рефлексивное и транзитовное замыкание.

Задания III.2.1:

1. Определите разницу между стековым автоматом и конечным автоматом.
2. Перечислите элементы стекового автомата.
3. Нарисуйте структуру стекового автомата.
4. Опишите функцию внутренней памяти стекового автомата.
5. Опишите функцию внешней памяти стекового автомата.
6. Приведите формальное определение стекового автомата.
7. Опишите функцию переходов стекового автомата.
8. Опишите такт работы стекового автомата.
9. Найтие рефлексивное и транзитовное замыкание отношения $\models$.

Вопросы III.2.1:

1. По какому принципу действует память стекового автомата?
2. Как работает стековый автомат?
3. Что такой такт стекового автомата?
4. Чем отличается недетерминированный стековый автомат от детерминированного?
5. Может ли стек содержать пустую цепочку?

6. Какой автомат мощнее между конечным автоматом и стековым автоматом?

7. Что означает запись (q_f, ε, η)?

III.2.2. Цепочки и языки, распознаваемые стековыми автоматами

В этом параграфе будут рассмотрены недетерминированные стековые автоматы, а также будут предложены примеры, даны задания, сформулированы вопросы [1-13,15-21,25,31].

Определение III.2.2.1. Входная цепочка $\tau \in \boldsymbol{T}^*$ распознается стековым автоматом $\boldsymbol{M}$:

1) при помощи финального состояния, если для некоторых $q_i \in \boldsymbol{F}$ ($i \geq 1$) и $h_0 \in \boldsymbol{H}$, $\eta \in \boldsymbol{H}^*$ выполняется отношение $(q_s,\tau,h_0) \models^* (q_i,\varepsilon,\eta)$;

2) при пустом стеке, если для некоторого состояния $q \in \boldsymbol{Q}$ выполняется $(q_s,\tau, h_0) \models^* (q, \varepsilon, \varepsilon)$.

Определение III.2.2.3. *Я*зыком $\boldsymbol{L(M)}$, распознаваемый стековым автоматом $\boldsymbol{M}$, называется множество распознанных этим автоматом входных цепочек.

Распознаваемый стековыми автоматами $\boldsymbol{M}$ язык $\boldsymbol{L}(\boldsymbol{M})$ определяется двумя способами:

1) при помощи финального состояния как

$$\boldsymbol{L(M)} = \{\tau:\ \tau \in \boldsymbol{T}^* \ \&\ (q_s, \tau, h_0) \models^* (q_f, \varepsilon, \eta) \ \&\ \eta \in \boldsymbol{H}^*\}$$

2) при пустом стеке как

$$\boldsymbol{L(M)} = \{\tau:\ \tau \in \boldsymbol{T}^* \ \&\ (q_s, \tau, h_0) \models^* (q, \varepsilon, \varepsilon) \ \&\ q \in \boldsymbol{Q}\}$$

Если язык распознается автоматом при пустом стеке, то его множество заключительных состояний не используется и в этом случае обычно принимается $\boldsymbol{F} = \varnothing$.

Для стековых автоматов класс языков, распознаваемых при помощи финального состояния, совпадает с классом языков, распознаваемых при пустом стеке.

Примеры III.2.2.1. Пусть задан язык $\boldsymbol{L}_1=\{0^n1^n:\ n \geq 0\}$. Построим стековый автомат $\boldsymbol{M}_1 = \langle \boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta \rangle$ такой, чтобы $\boldsymbol{L}_1=\boldsymbol{L}(\boldsymbol{M}_1)$. Для этого положим $\boldsymbol{Q} = \{q_0, q_1, q_2\}$, $\boldsymbol{T} = \{0,1\}$, $q_s = q_0$, $\boldsymbol{F} = \{q_0\}$, $\boldsymbol{H} = \{h,0\}$, $h_0 = h$ и

$$\delta(q_0, 0, h) = \{(q_1,0h)\},$$
$$\delta(q_1,0,0) = \{(q_1,00h)\},$$

$$\delta(q_1,1,0) = \{(q_2,0h)\},$$
$$\delta(q_2,1,0) = \{(q_2,h)\},$$
$$\delta(q_2,\varepsilon, h) = \{(q_0,\varepsilon)\}.$$

Работа этого автомата состоит в том, что он копирует в стеке начальную часть входной цепочки, состоящую из нулей, а затем устраняет из стека по одному нулю на каждую единицу, которую он видит на входной цепочке. Кроме того, переходы состояний гарантируют, что все нули предшествуют единицам. Например, для входной цепочки 0011 автомат $\boldsymbol{M}_1$ совершает такую последовательность тактов:

$$(q_0,0011,h) \models (q_1,011,0h) \models$$
$$(q_1,11,00h) \models (q_2,1,0h) \models$$
$$(q_2,\varepsilon,h) \models (q_0,\varepsilon,\varepsilon)$$

1) Доказательство отношения $\boldsymbol{L}_1 \subseteq \boldsymbol{L}(\boldsymbol{M}_1)$. В общем случае можно показать, что

$$(q_0, 0, h) \models (q_1, \varepsilon, 0h)\ ) \models (q_1, 0^i, 0h) \models^i (q_1, \varepsilon, 0^{i+1}h)\ ) \models$$
$$(q_1, 1, 0^{i+1}h) \models (q_2, \varepsilon, 0^i h)\ ) \models (q_2, 1^i, 0^i h) \models^i (q_2, \varepsilon, h)$$
$$(q_2, e, h) \models (q_0, \varepsilon, \varepsilon)$$

Объединяя все это, получаем для $n \geq 1$

$$(q_0, 0^n 1^n, h) \models^{2n+1} (q_0, \varepsilon, \varepsilon) \text{ и } (q_0, \varepsilon, h) \models (q_0, \varepsilon, h)$$

Таким образом, $\boldsymbol{L}_1 \subseteq \boldsymbol{L}(\boldsymbol{M}_1)$.

2) Доказательство $\boldsymbol{L}_1 \supseteq \boldsymbol{L}(\boldsymbol{M}_1)$. Положим, что $\boldsymbol{M}_1$ распознает только цепочки вида 0^n1^n. Эта часть доказательства труднее. Обычно легче доказать, что такие-то цепочки распознаватель распознает, и так же, как для грамматики, труднее доказать, что она порождает цепочки только определенного вида.

Заметим, что если $\boldsymbol{M}_1$ распознает непустую цепочку, то он должен пройти через состояния в таком порядке q_0, q_1, q_2, q_0.

Далее, если $(q_0, \tau, h) \models^i (q_1, \varepsilon, \eta)$ для $i \geq 1$, то $\tau = 0^i$ и $\eta = 0^i h$. Аналогично, если $(q_2, \tau, \eta) \models^i (q_2, \varepsilon, \beta)$, то $\tau = 1^i$ и $\eta = 0^i\beta$. К тому же $(q_1,\tau,\eta) \models (q_2,\varepsilon,\beta)$ только тогда, когда $\tau = 1$ и $\eta = 0\beta$, а $(q_2,\tau,h) \models^* (q_0,\varepsilon,\varepsilon)$ только тогда, когда $\tau = \varepsilon$. Таким образом, если $(q_0, \tau, h) \models^i (q_0, \varepsilon, \eta)$

для некоторого $i \geq 0$, то либо $\tau = \varepsilon$, $i = 0$, либо $\tau = 0^n 1^n$, $i = 2n+1$, $\eta = \varepsilon$. Следовательно, $\boldsymbol{L}_1 \supseteq \boldsymbol{L}(\boldsymbol{M}_1)$.

Наконец, из двух доказанных вспомогательных утверждений заключаем, что $\boldsymbol{L}_1 = \boldsymbol{L}(\boldsymbol{M}_1)$. Иначе говоря, класс языков, распознаваемых при помощи финального состояния, совпадает с классом языков, распознаваемых при пустом стеке.

Примеры III.2.2.2. Следующий стековый автомат распознает язык $\{xCx' : x \in \{0, 1\}^*\}$ при пустом стеке:

$$\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>, \text{ где}$$

где $\boldsymbol{T} = \{0, 1, c\}$, $\boldsymbol{Q} = \{q_1, q_2\}$, $H = \{R, B, D\}$, $q_0 = q_1$, $h_0 = R$ и

$\delta(q_1, 0, R) = \{(q_1, BR)\}$, $\delta(q_2, 0, B) = \{(q_2, \varepsilon)\}$,

$\delta(q_1, 0, B) = \{(q_1, BB)\}$, $\delta(q_2, 1, D) = \{(q_2, \varepsilon)\}$,

$\delta(q_1, 0, D) = \{(q_1, BD)\}$, $\delta(q_2, \varepsilon, R) = \{(q_2, \varepsilon)\}$,

$\delta(q_1, c, R) = \{(q_2, R)\}$, $\delta(q_1, 1, R) = \{(q_1, DR)\}$,

$\delta(q_1, c, B) = \{(q_2, B)\}$, $\delta(q_1, 1, B) = \{(q_1, DB)\}$,

$\delta(q_1, c, D) = \{(q_2, D)\}$, $\delta(q_1, 1, D) = \{(q_1, DD)\}$.

Пусть $x = 001$, тогда $xcx^T = 001c100$. Последовательность переходов для входной цепочки 001c100 такова:

$$(q_1, R) \models^0 (q_1, BR) \models^0$$
$$(q_1, BBR) \models^1 (q_1, DBBR) \models^c$$
$$(q_2, DBBR) \models^1 (q_2, BBR) \models^0$$
$$(q_2, BR) \models^0 (q_2, R) \models^\varepsilon (q_2, \varepsilon).$$

Теперь мы определим некоторые варианты стековых автоматов и установим связь между определяемыми ими языками и языками, распознаваемыми обычными стековыми автоматами.

Определение III.2.2.4. *Расширенным стековым автоматом* назовем следующую десятку $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$, где δ – отображение конечного подмножества множества $\boldsymbol{Q} \times (\boldsymbol{T} \cup \{\varepsilon\}) \times \boldsymbol{H}^*$ в множество всех конечных подмножеств множества $\boldsymbol{Q} \times \boldsymbol{H}^*$, т.е. δ: $\boldsymbol{Q} \times (\boldsymbol{T} \cup \{\varepsilon\}) \times \boldsymbol{H}^* \rightarrow \mathcal{P}(\boldsymbol{Q} \times \boldsymbol{H}^*)$, а все другие символы имеют тот же смысл, что и раньше.

Для расширенного стекового автомата $\boldsymbol{M}$ конфигурация и язык, распознаваемым этим автоматом $\boldsymbol{M}$ определяются также как и в обычном недетерминированном стековом автомате.

Класс языков, распознаваемых расширенным СА, совпадает с классом языков, распознаваемых обычным НСА. Но в отличие от обычного НСА расширенный СА может продолжать работу и тогда, когда стек пуст.

Пусть $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$ – недетерминированный стековый автомат или расширенный стековый автомат.

Определение III.2.2.5. Говорят, что $\boldsymbol{M}$ распознает цепочку $\tau \in \boldsymbol{T}^*$ опустошением стека, если выполняется $(q_s, \tau, h_0) \models^+ (q, \varepsilon, \varepsilon)$ для некоторого $q \in \boldsymbol{Q}$.

Определение III.2.2.6. *Я*зыком $\boldsymbol{L_\varepsilon(M)}$, распознаваемый стековым автоматом опустошением стека $\boldsymbol{M}$, называется множество распознанных цепочек $\tau \in \boldsymbol{T}^*$ опустошением стека, т.е.

$$\boldsymbol{L_\varepsilon(M)} = \{\tau: \ \tau \in \boldsymbol{T}^* \ \& \ (q_s, \tau, h_0) \models^+ (q, \varepsilon, \varepsilon) \ \& \ q \in \boldsymbol{Q}\}$$

Класс языков, распознаваемых стековым автоматом опустошением стека, совпадает с классом языков, распознаваемых обычным недетерминированным стековым автоматом, т.е. для заданного стекового автомата $\boldsymbol{M'}$ можно всегда построить такой стековый автомат $\boldsymbol{M}$, что $\boldsymbol{L(M) = L_\varepsilon(M')}$.

Недетерминированный стековый автомат обеспечивает удобную абстракцию определения языка, но для практической реализации его надо моделировать (реализовать программно или аппаратно) в детерминированном виде. При этом нужно учесть, что функциональные возможности недетерминированных и детерминированных стековых автоматов разные.

Задания III.2.2:

1. Постройте СА, распознающий язык $\{\omega\omega^R: \ \omega \in \{a,b\}\}$
2. Постройте СА, распознающий язык $\{0^n1^n: \ n \geq 0\}$. Пусть задан $\boldsymbol{M} = <\{q_0, q_1, q_2\}, \{0, 1\}, q_0, \{q_0\}, \{Z, a\}, Z, \delta>$. Входная цепочка

000111.

3. Постройте СА, распознающий язык $\boldsymbol{L}=\{a^n b^n : n\geq 0\}$. Пусть $\boldsymbol{M} = <\{q_0,q_1,q_2\}, \{a, b\}, q_0, \{q_0\}, \{Z, a\}, Z, \delta>$. Входная цепочка *aabb*.

4. Определите расширенный СА $\boldsymbol{M}$, распознающий язык $\{\omega\omega^h\varepsilon\{k,\boldsymbol{L}\}^*\}$, например, для входа *kkllkk*.

5. Пусть $\boldsymbol{M} = <\{q, p\}, \{k,\boldsymbol{L}\}, q, \{p\}, \{k, l, H, Z\}, Z, \delta>$, а функция δ определяется следующим образом:

$$\delta(q,k,\varepsilon) = \{(q,\mathrm{k})\},$$
$$\delta(q,\varepsilon,\varepsilon) = \{(q,\varepsilon)\},$$
$$\delta(q,\varepsilon,\varepsilon) = \{(q,H)\},$$
$$\delta(q,\varepsilon,kHk) = \{(q,H)\},$$
$$\delta(q,\varepsilon,kHk) = \{(q,H)\},$$
$$\delta(q,\varepsilon,HZ) = \{(p,k)\}.$$

6. Постройте СА, определяющий язык $\{(01)^n : n\geq 0\}$. Пусть $\boldsymbol{M} = <\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{0, 1\}, \{Z,0\}, \delta, q_0, Z, \{q_0\}>$. Ввходная цепочка 010101.

7. Постройте СА, распознающий язык, который порождается БГ с правилами $S\to aSbS$, $S\to bSaS$, $S\to\varepsilon$.

Вопросы III.2.2:

1. Какие виды конфигурации имеются в стековом автомате и как они определяются?

2. Что такой язык, распознаваемый недетерминированным стековым автоматом?

3. Какая связь имеется между языками, распознаваемыми расширенным стековым автоматом, и обычным недетерминированным стековым автоматом?

4. Что такой язык, распознаваемый стековым автоматом при помощи финального состояния?

5. Что такой язык, распознаваемый стековым автоматом опустошением стека?

6. Может ли СА распознать язык:

$$\{\omega \in \{a, b\}^*: |\omega|_a = |\ \omega\ |_b\}.$$

III.2.3. Детерминированные стековые автоматы

В этом параграфе будет рассмотрено формальное определение детерминированного стекового автомата, а также будут предложены примеры, даны задания, сформулированы вопросы [1-13,15-21,25,31].

В практических задачах нас больше интересуют детерминированные стековые автоматы (ДСА), т. е. такие, которые в каждой конфигурации могут сделать не более одного очередного такта. В этом параграфе мы займемся ДСА и в дальнейшем увидим, что, к сожалению, ДСА не так мощны по своей распознавательной способности, как НСА. Существуют КС-языки, которые нельзя определить ДСА.

Язык, распознаваемый детерминированным стековым автоматом, называется детерминированным КС-языком.

Определение III.2.3.1. Заданный стековый автомат $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$ называется детерминированным стековым автоматом (ДСА), если для каждых $q \in \boldsymbol{Q}$ и $h \in \boldsymbol{H}$ либо

(1) $\delta(q, t, h)$ содержит не более одного элемента для каждого $t \in \boldsymbol{T}$ и $\delta(q, \varepsilon, h) = \varnothing$, либо

(2) $\delta(q, t, h) = \varnothing$ для всех $t \in \boldsymbol{T}$ и $\delta(q, \varepsilon, h)$ содержит не более одного элемента.

В силу этих ограничений ДСА в любой конфигурации может выбрать не более одного такта. Это позволит гораздо легче моделировать ДСА, чем НСА.

Соглашение III.2.3. В силу того, что функция перехода ДСА $\delta(q, t, h)$ содержит не более одного элемента, будем писать $\delta(q, t, h) = (h, \gamma)$ вместо $\delta(q, t, h) = \{(h, \gamma)\}$.

Определение III.2.3.2. Расширенный стековый автомат (РСА) $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$ называется детерминированным, если выполнены следующие условия:

(1) $\delta(q, a, \eta) \le 1$ для всех $q \in \boldsymbol{Q}$, $a \in \boldsymbol{T} \cup \{\varepsilon\}$ и $\eta \in \boldsymbol{H}^*$;

(2) если $\delta(q, a, \eta) \neq \varnothing$, $\delta(q, a, \xi) \neq \varnothing$ и $\eta \neq \xi$, то ни одна из цепочек η и ξ не является постфиксом другой;

(3) если $\delta(q, a, \eta) \neq \varnothing$ и $\delta(q, \varepsilon, \xi) \neq \varnothing$, то ни одна из цепочек η и ξ не является постфиксом другой.

Сначала модифицируем детерминированный стековый автомат так, чтобы для любой конфигурации, в которой часть входа осталась непрочитанной, был всегда возможен очередной такт. Следующая лемма показывает, как это сделать.

Лемма III.2.3.1. Пусть $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$ – детерминированный стековый автомат. Тогда можно построить такой эквивалентный детерминированный стековый автомат

$$\boldsymbol{M}' = <\boldsymbol{Q}', \boldsymbol{T}, q'_s, \boldsymbol{F}, \boldsymbol{H}', h'_0, \vdash, \dashv, \perp, \delta'>,$$ что:

(1) для всех $a \in \boldsymbol{T}$, $q \in \boldsymbol{Q}'$ и $h \in \boldsymbol{H}'$ либо $\delta'(q, t, h)$ содержит точно один элемент и $\delta'(q, \varepsilon, h) = \varnothing$, либо $\delta'(q, t, h) = \varnothing$ и $\delta'(q, \varepsilon, h)$ содержит точно один элемент;

(2) если $\delta'(q, t, h'_0) = (h, \gamma)$ для некоторого $a \in \boldsymbol{T} \cup \{\varepsilon\}$, то $\gamma = \alpha h'_0$ для некоторой цепочки $\alpha \in \boldsymbol{H}^*$.

Определение III.2.3.3. Конфигурация (q, ω, γ) ДСА $\boldsymbol{M}$ называется зацикливающей, если для каждого $i \ge 1$ найдется конфигурация (p_i, ω, β_i),что $|\beta| \ge |\gamma|$ и

$$(q, \omega, \gamma) \models (p_i, \omega, \beta_1) \models (p_2, \omega, \beta_2) \models \ldots$$

Таким образом, конфигурация зацикливающая, если $\boldsymbol{M}$ может делать бесконечное число ε-тактов, не укорачивая стек; при этом стек может либо бесконечно расти, либо циклически совпадать с несколькими различными цепочками.

Заметим, что существуют незацикливающие конфигурации, которые после ряда ε-тактов, укорачивающих стек, переходят в зацикливающую конфигурацию. Нельзя сделать бесконечное число ε-тактов, исходя из любой данной конфигурации, не попав через конечное число тактов в зацикливающую конфигурацию.

Если $\boldsymbol{M}$ попадает в зацикливающую конфигурацию в середине входной цепочки, то он больше не будет использовать входную цепочку. Мы хотим преобразовать данный детерминированный стековый автомат $\boldsymbol{M}$ в эквивалентный детерминированный стековый автомат $\boldsymbol{M}'$, который никогда не попадает в зацикливающую конфигурацию. Для этого нужно обнаружить зацикливающих конфигураций в стековом автомате. Следовательно, нужно построить следующий алгоритм:

Алгоритм III.2.3. Обнаружение зацикливающих конфигураций

Вход. ДСА $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$.

Выход. Для всех $q \in \boldsymbol{Q}$ и $h \in \boldsymbol{H}$ имеется два выхода:

(1) $C_1 = \{(q, h) :$ (q, ε, h) – зацикливающая конфигурация и не существует такого $p \in \boldsymbol{F}$, что $(q, \varepsilon, h) \models^* (p, \varepsilon, \gamma)$ для некоторой цепочки $\gamma \in \boldsymbol{H}^*\}$ и

(2) $C_2 = \{(q, h) :$ (q, ε, h) – зацикливающая конфигурация и $(q, \varepsilon, h) \models^* (p, \varepsilon, \gamma)$ для некоторых $p \in \boldsymbol{F}$ и $\gamma \in \boldsymbol{H}^*\}$.

Метод. Пусть $\#\boldsymbol{Q}=n_1$, $\#\boldsymbol{H}=n_2$ и 1 – длина самой длинной цепочки, которую ДСА $\boldsymbol{M}$ может записать в стек за один такт. Пусть $n_3 = n_1(n^{n_1}{}_2{}^{n_2}{}^{l+\boldsymbol{L}} - n_2) / (n_2 - 1)$, если $n_2 > 1$, и $n_3 = n_1$, если $n_2 = 1$. Число n_3 – это максимальное число ε-тактов, которое может сделать $\boldsymbol{M}$, не зацикливаясь.

(1) Для всех $q \in \boldsymbol{Q}$ и $h \in \boldsymbol{H}$ выяснить, выполняется ли $(q, \varepsilon, h) \models^{n_3} (p, \varepsilon, \gamma)$ для каких-нибудь $p \in \boldsymbol{Q}$ и $\gamma \in \boldsymbol{H}^+$. При этом используется прямое моделирование $\boldsymbol{M}$. Если да, то (q, ε, h) – зацикливающая конфигурация, так как в этом случае – мы это покажем – должна быть такая пара (q', h'), где $q' \in \boldsymbol{Q}$ и $h' \in \boldsymbol{H}$, что

$$(q, \varepsilon, h) \models^* (q', \varepsilon, h'\beta) \models^m (q', \varepsilon, h'\gamma\beta) \models^{m(j-1)} (q', \varepsilon, h'\gamma^j\beta)$$

где $m>0$ и $j>0$, $\gamma^j \in \boldsymbol{H}^*$, $\beta \in \boldsymbol{H}^*$. Заметим, что γ может быть ε.

(2) Если (q, ε, h) – зацикливающая конфигурация, выяснить, существует ли такое $p \in \boldsymbol{F}$, что $(q, \varepsilon, h) \models^j (p, \varepsilon, \gamma)$ для некоторого $0 \le j \le n_3$. При этом снова используется прямое моделирование. Если да, то включить (q, h) в C_2. В противном случае включить (q, h) в C_1. Мы утверждаем, что если $\boldsymbol{M}$ может достичь заключительной

конфигурации, исходя из (q, ε, h), то это должно произойти за n_3 или меньшее число тактов.

Алгоритм III.2.3 правильно находит множества C_1 и C_2.

Определение III.2.3.4. Детерминированный стековый автомат $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$ назовем дочитывающим, если для каждой цепочки $\omega \in \boldsymbol{T}^*$ существуют такие $p \in \boldsymbol{Q}$ и $\gamma \in \boldsymbol{H}^*$, что с верно $(q_0, \omega, h_0) \models^* (p, \varepsilon, \gamma)$. Дочитывающим называется ДСА, способный дочитывать до конца каждую входную цепочку.

Лемма III.2.3.2. Пусть $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$ – ДСА. Тогда существует эквивалентный ему дочитывающий ДСА $\boldsymbol{M}'$.

Теорема III.2.3.1. Если $\boldsymbol{L}=\boldsymbol{L}(\boldsymbol{M})$ для некоторого детерминированного стекового автомата $\boldsymbol{M}$, то $\boldsymbol{L}=\boldsymbol{L}(\boldsymbol{M}')$ для некоторого детерминированного стекового автомата $\boldsymbol{M}'$.

Если для конечных автоматов детерминированные и недетерминированные модели эквивалентны по отношению к классу распознаваемых языков, то этого нельзя сказать для стековых автоматов. Детерминированные стековые автоматы допускают лишь некоторое подмножество бесконтекстных языков, которые называют детерминированными бесконтекстными языками.

Примеры III.2.3. *Я*зык $\{xx^T: x \in \{0,1\}^*\}$ распознается НСА $\boldsymbol{M}$. Пусть $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_0, \varnothing, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$, где $\boldsymbol{Q} = \{q_1, q_2\}$, $\boldsymbol{T} = \{0,1\}$, $\boldsymbol{H} = \{A, B, C\}$, $q_0 = q_1$, $h_0 = A$, $\lambda \in \boldsymbol{H}^*$ и δ:

$$\delta(q_1, 0, A) = \{(q_1, BA)\},$$
$$\delta(q_1, 1, C) = \{(q_1, CC), (q_2, \lambda)\},$$
$$\delta(q_1, 1, A) = \{(q_1, CA)\},$$
$$\delta(q_2, 0, B) = \{(q_2, \lambda)\},$$
$$\delta(q_1, 0, B) = \{(q_1, BB), (q_2, \lambda)\},$$
$$\delta(q_2, 1, C) = \{(q_2, \lambda)\},$$
$$\delta(q_1, 0, C) = \{(q_1, BC)\},$$
$$\delta(q_1, \lambda, A) = \{(q_2, \lambda)\},$$

$\delta(q_1, 1, B) = \{(q_1, CB)\}$,

$\delta(q_2, \lambda, A) = \{(q_2, \lambda)\}$.

Но, язык $\{xxT : x \in \{0.1\}^*\}$ не распознается никаким детерминированным стековым автоматом.

Задания III.2.3:

1. Для распознавания языка $\boldsymbol{L}=\{(01)^n: n \geq 0\}$ построить ДСА $\boldsymbol{M}=<\{q_0,q_1,q_2,q_3,q_4,q_5,q_6\},\{0,1\},q_0,\{q_0\},\{Z,0\},Z,\delta>$, вход 010101.

2. Построить ДСА $\boldsymbol{M} = <\{q_0,q_1,q_2\},\{a,b\},q_0, \{q_0\}, \{Z,a\}, Z, \delta>$, распознающий язык $\{a^n b^n : n \geq 0\}$, вход $aabb$.

3. Построить ДСА = <{{q_0,q_1,q_2},{‘{’,‘}’},q_0,{q_0},{Z,’{‘},Z,δ>, читающий количество скобок в арифметическом выражении.

4. Пусть расширенный ДСА $\boldsymbol{M} = <\{q, r\}, \boldsymbol{T}, q, \{r\}, \$, H, \delta>$, где вход a & $a \cap a$, а функция δ определяется так:

(1) $\delta(q,b,\varepsilon)=\{(q,b)\}$ для всех $b \in \{a, +, *, (,), \&, \cap\}$;

(2) $\delta(q, \varepsilon, T \cap F) = \{(q, E)\}$, $\delta(q, \varepsilon, T) = \{(q, E)\}$, $\delta(q, \varepsilon, T \cap F) = \{(q, T)\}$,
$\delta(q, \varepsilon, F) = \{(q, T)\}$, $\delta(q, \varepsilon, (E)) = \{(q, F)\}$, $\delta(q, \varepsilon, a) = \{(q, F)\}$;

(3) $\delta(q, \varepsilon, \$E) = \{(r, \varepsilon)\}$.

5. Построить ДСА $\boldsymbol{M} = <\{q, p\},\{0,1\}, q,\{p\}, Z, \{0,1,X, Z\}, \delta>$, распознающий язык $\{\omega\omega^r \varepsilon \{0,1\}^*\}$, на вход 001100, а функция

$\delta(q,0,\varepsilon)=\{(q,0)\}$,

$\delta(q,c,\varepsilon)=\{(q,c)\}$,

$\delta(q,\varepsilon,\varepsilon)=\{(q,X)\}$,

$\delta(q,\varepsilon,0X0)=\{(q,X)\}$,

$\delta(q,a,cXc)=\{(q,X)\}$,

$\delta(q,\varepsilon,XZ)=\{(p,0)\}$.

Вопросы III.2.3:

1. Какие условия накладываются на расширенный стековый автомат, чтобы он был детерминированным?

2. При каких условиях конфигурация детерминированного называется зацикливающей?

3. При каких условиях детерминированный стековый автомат становится дочитывающим автоматом?

III.3. Свойства бесконтекстных языков

III.3.1. Эквивалентность беконтекстных грамматик и стековых автоматов

В этом параграфе будет обсуждена эквивалентность бесконтекстных грамматик и стековых автоматов (с магазинной памятью), а также будут предложены примеры, даны задания, сформулированы вопросы [1-4, 6-9, 11-13, 15-21, 24, 27-34].

Можно показать, что класс языков, порождаемых бесконтекстными грамматиками (БГ) совпадает с классом языков, распознаваемых стековыми автоматами (СА). Это доказывается построением НСА, соответствующего данной БГ, и наоборот, построением БГ, соответствующей данному СА.

Определение III.3.1.1. КС-грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ называется без ε-*правил* (или неукорачивающей), если:

1) либо, множество $\boldsymbol{P}$ не содержит ε-правил,
2) либо, имеется только одно ε-правило $S \to \varepsilon$ и нетерминал S не всречается во множестве $\boldsymbol{P}$.

Определение III.3.1.2. *Расширенным стековым автоматом* называют следующую десятку $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$, где δ – отображает множество $\boldsymbol{Q}\times(\boldsymbol{T}\cup\{\varepsilon\})\times\boldsymbol{H}^*$ во множество всех подмножеств множества $\boldsymbol{Q}\times\boldsymbol{H}^*$, т.е. δ: $\boldsymbol{Q}\times(\boldsymbol{T}\cup\{\varepsilon\})\times\boldsymbol{H}^* \to \mathcal{I}(\boldsymbol{Q}\times\boldsymbol{H}^*)$, $\mathcal{I}(\boldsymbol{Q}\times\boldsymbol{H}^*)$ – множество всех подмножеств множества $\boldsymbol{Q}\times\boldsymbol{H}^*$, а значения остальных символов остаются прежними.

Для расширенного СА конфигурация и язык, распознаваемый этим автоматом, определяется как в обычном СА.

Лемма III.3.1.1. Если $\boldsymbol{L}(\boldsymbol{G})$ – контекстно-свободный язык, порождаемый БГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ в нормальной форме Грейбах, то существует недетерминированный СА $\boldsymbol{M}$, такой, что $\boldsymbol{L}(\boldsymbol{M}) = \boldsymbol{L}(\boldsymbol{G})$.

Доказательство. Пусть задан следующий стековый автомат

$\boldsymbol{M}$ = <$\boldsymbol{Q}$, $\boldsymbol{T}$, q_s, $\boldsymbol{F}$, $\boldsymbol{H}$, h_0, $\vdash$,$\dashv$, $\perp$, δ> – СА, где δ: $\boldsymbol{Q}\times(\boldsymbol{T}\cup\{\varepsilon\})\times\boldsymbol{H}\to\mathcal{P}(\boldsymbol{Q}\times\boldsymbol{H}^*)$, $\boldsymbol{Q}=\{q\}$, $\boldsymbol{T}=\boldsymbol{T}$, $q_S=q$, $\boldsymbol{F}=\varnothing$, $\boldsymbol{H}=\boldsymbol{N}\cup\boldsymbol{T}$, $h_0=S$, а функция перехода δ определяется как:

(1) $\delta(q, t, A)=\{(q, t)\}$ тогда и только тогда, когда $A\to t\tau\in\boldsymbol{P}$ для некоторых $A\in\boldsymbol{N}$, $t\in\boldsymbol{T}$, $\tau\in\boldsymbol{N}^*$;

(2) $\delta(q, t, t)=\{(q, \varepsilon)\}$ для всех $t\in\boldsymbol{T}$.

Мы хотим показать, что $A\Rightarrow^m\omega$ тогда и только тогда, когда $(q, \omega, A)\models^n(q, \varepsilon, \varepsilon)$ для некоторых $m\geq1$ и $n\geq1$, т.е.

$$\exists m(m\geq1 \;\&\; A\Rightarrow^m\omega) \Leftrightarrow \exists n(n\geq1 \;\&\; (q, \omega, A)\models^n(q, \varepsilon, \varepsilon)) \quad \text{(III.3.1.1)}$$

Необходимость условия доказывается индукцией по m.

Базис индукции. Допустим, что $A\Rightarrow^m\omega$. Если $m=1$ и $\omega=t_1\ldots t_k$ ($k\geq0$), то $(q, t_1\ldots t_k, A)\models(q, t_1\ldots t_k, t_1\ldots t_k)\models^k(q, \varepsilon, \varepsilon)$.

Индукционный шаг. Предположим, что $A\Rightarrow^m\omega$ для некоторого $m>1$. Первый шаг этого вывода должен иметь вид $A\Rightarrow X_1\ldots X_k$, где $X_i\Rightarrow^{m_i}x_i$ для некоторого $m_i<m$, $1\leq i\leq k$ и $x_1\ldots x_k=\omega$. Тогда $(q, \omega, A)\models(q, \omega, X_1\ldots X_k)$.

Если $X_i\in\boldsymbol{N}$, то по предположению индукции $(q,x_i,X_i)\models^*(q,\varepsilon,\varepsilon)$. Если $X_i=x_i\in\boldsymbol{T}$, то $(q, x_i, X_i)\models(q, \varepsilon, \varepsilon)$.

Объединяя вместе эти последовательности тактов, можно получить $(q, \omega, A)\models^+(q, \varepsilon, \varepsilon)$.

Достаточность условия доказывается индукцией по n.

Базис индукции. Допустим, что $(q, \omega, A)\models^n(q, \varepsilon, \varepsilon)$. Если $n=1$, то $\omega=\varepsilon$ и $A\to\varepsilon\in\boldsymbol{P}$, т.е. доказана, что если $(q, \omega, A)\models^+(q, \varepsilon, \varepsilon)$, то $A\Rightarrow^+\omega$.

Индукционный шаг. Предположим, что $(q, \omega, A)\models^{n_i}(q, \varepsilon, \varepsilon)$ для всех $n_i<n$. Тогда первый такт, сделанный СА-ом $\boldsymbol{M}$, должен иметь вид $(q, \omega, A)\models(q, \omega, X_1\ldots X_k)$ причем $(q, x_i, X_i)\models^{n_i}(q, \varepsilon, \varepsilon)$ для $1\leq i\leq k$ и $\omega=x_1\ldots x_k$. Тогда $A\to X_1\ldots X_k\in\boldsymbol{P}$, по предложению индукции $X_i\Rightarrow^+x_i$ для $X_i\in\boldsymbol{N}$ и $X_i\Rightarrow^0x_i$ для $X_i\in\boldsymbol{T}$. Таким образом, вывод цепочки ω из A в грамматике $\boldsymbol{G}$ можно представить как:

$$A\Rightarrow X_1\ldots X_k$$
$$\Rightarrow^* x_1\ldots X_{k-1}X_k$$

$$\ldots\ldots$$
$$\Rightarrow^* x_1 \ldots x_{k-1}X_k$$
$$\Rightarrow^* x_1 \ldots x_{k-1}x_k = \omega$$

Из (III.3.1.1), в частности, вытекает, что $S\Rightarrow^+\omega$ тогда и только тогда, когда $(q, \omega, S) \models^+ (q, \varepsilon, \varepsilon)$. Следовательно, $\boldsymbol{L}(\boldsymbol{M}) = \boldsymbol{L}(\boldsymbol{G})$.

Лемма III.3.1.2. Пусть задан некоторый $\boldsymbol{L}(\boldsymbol{M})$ – язык, распознаваемый СА $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>$. Тогда существует такая БГ $\boldsymbol{G}$, что $\boldsymbol{L}(\boldsymbol{G}) = \boldsymbol{L}(\boldsymbol{M})$.

Доказательство. Построим грамматику $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ так:

1) $\boldsymbol{N} = \{(q, A, p)\}$, $q\in\boldsymbol{Q}$, $p\in\boldsymbol{Q}$, $A\in\boldsymbol{H}\cup\{S\}$;

2) $\boldsymbol{T} = \boldsymbol{T}$;

3) $S \rightarrow (q_0, x_0, q) \in \boldsymbol{P}$ при любом $q\in\boldsymbol{Q}$;

4) $(q, A, p)\rightarrow a(q_1, B_1, q_2)(q_2, B_2, q_3)\ldots (q_m, B_m, q_{m+1})\in\boldsymbol{P}$ для любых $q = q_1, q_2,\ldots, q_{m+1} = p\in\boldsymbol{Q}$, любого $a\in(\boldsymbol{T}\cup\{\varepsilon\})$ и любых $A, B_1, B_2,\ldots, B_m\in\boldsymbol{H}$, таких, что $q_1, B_1, B_2,\ldots, B_m \in\delta(q, a, A)$. (Если m=0, то $q_1 = p(p\varepsilon)\in\delta(q, a, A)$ и $(q, A, p)\rightarrow a \in \boldsymbol{P}$).

Теорема III.3.1. Класс языков, порождаемых контекстно-свободными грамматиками, в точности совпадает с классом языков, распознаваемых стековыми автоматами.

Доказательство следует из Леммы III.3.1.1 и Леммы III.3.1.2.

Примеры III.3.1.1. Пусть задана БГ $\boldsymbol{G} = <\boldsymbol{N},\boldsymbol{T},\boldsymbol{P},S>$, где $\boldsymbol{N}=\{S,A,B\}$, $\boldsymbol{T} =\{a,b\}$ и множество правил вывода имеет следующий вид:

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

Можно построить следующий недетерминированный СА $\boldsymbol{M}$ так: $\boldsymbol{Q} = \{q_1\}$, $\boldsymbol{T} = \{a, b\}$, $\boldsymbol{F} = \varnothing$, $q_s = q_1$, $\boldsymbol{H} = \{S, A, B\}$, $h_0 = S$ и δ:

$\delta(q_1, a, S) = \{(q_1, B)\}$, поскольку $S\rightarrow aB \in \boldsymbol{P}$;

$\delta(q_1, b, S) = \{(q_1, A)\}$, поскольку $S\rightarrow bA \in \boldsymbol{P}$;

$\delta(q_1, a, A) = \{(q_1, S), (q_1\varepsilon)\}$, поскольку $A\rightarrow aS \in \boldsymbol{P}$ и $A\rightarrow a \in \boldsymbol{P}$;

$\delta(q, b, A) = \{(q_1, AA)\}$, поскольку $A\rightarrow bAA \in \boldsymbol{P}$;

$\delta(q_1, aB) = \{(q_1, BB)\}$, поскольку $B\rightarrow aBB \in \boldsymbol{P}$;

$\delta(q_1, bB) = \{(q_1, S), (q_1,\varepsilon)\}$, поскольку $B \to bS \in \boldsymbol{P}$ и $B \to b \in \boldsymbol{P}$;

По теореме 1 $\boldsymbol{L}(\boldsymbol{M}) = \boldsymbol{L}(\boldsymbol{G})$.

Класс языков, распознаваемых расширенными стековыми автоматами, совпадает с классом языков, распознаваемых обычными недетерминированными стековыми автоматами.

Лемма III.3.1.3. Если $\mathscr{L}(\boldsymbol{G})$ – контекстно-свободный язык, порождаемый грамматикой $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, то существует расширенный СА $\boldsymbol{M}$, такой, что $\boldsymbol{L}(\boldsymbol{M}) = \boldsymbol{L}(\boldsymbol{G})$.

Доказательство. Пусть задан стековый автомат

$$\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, q_s, \boldsymbol{F}, \boldsymbol{H}, h_0, \vdash, \dashv, \perp, \delta>, \text{ где}$$

$\boldsymbol{Q} = \{q, p\}$, $\boldsymbol{T} = \boldsymbol{T}$, $q_S = q$, $\boldsymbol{F} = \{p\}$, $\boldsymbol{H} = \boldsymbol{N} \cup \boldsymbol{T} \cup \{\$\}$, $h_0 = \$$, δ:

(1) $\delta(q, t, \varepsilon) = \{(q, t)\}$ для всех $t \in \boldsymbol{T}$ – на этих тактах входные символы переносятся в стек);

(2) Если $A \to \eta \in \boldsymbol{P}$, то $\delta(q,\varepsilon,\eta) = \{(q,A)\}$ для некоторых $A \in \boldsymbol{N}$ и $\eta \in \boldsymbol{H}^*$;

(3) $\delta(q, \varepsilon, \$S) = \{(p, \varepsilon)\}$.

Покажем, что процесс вычисления в автомате $\boldsymbol{M}$ заключается в построении правовыводимых цепочек грамматики $\boldsymbol{G}$, начиная с терминальной цепочки на входе $\boldsymbol{M}$ и кончая начальным нетерминалом S. Индукцией по n докажем, что

$$S \Rightarrow^* p\eta A,\ y \Rightarrow^n pxy \text{ влечет } (q, xy, \$) \models^* (q, y, \$\eta A) \quad \text{(III.3.1.2)}$$

Базис индукции. При $n=0$ $\boldsymbol{M}$ не делает ни одного такта.

Индукционный шаг. Предположим, что (III.3.1.2) верно для всех значений параметра, меньших n. Напишем $\eta A y \Rightarrow p\eta\beta y \Rightarrow^{n-1} pxy$. Допустим, что цепочка $\eta\beta$ состоит только из терминалов. Тогда $\eta\beta = x$ и $(q, xy, \$) \models^* (q, y, \$\eta\beta) \models (q, y, \$\eta A)$.

Если $\eta\beta \notin \boldsymbol{T}^*$, то можно писать $\eta\beta = \eta B h$, где B – самый правый нетерминал. По (III.3.1.2) из $S \Rightarrow^* p\eta B h y \Rightarrow^{n-1} pxy$ следует $(q, xy\$) \models^* (q, hy, \$\eta B)$. Кроме того, $(q, hy, \$\eta B) \models^* (q, y, \$\eta Bh) \models (q, y, \$\eta A)$ –тоже возможная последовательность тактов.

Заключаем, что (III.3.1.2) верно для всех n. Так как $(q, \varepsilon, \$S) \models (p, \varepsilon, \varepsilon)$, то $\boldsymbol{L}(\boldsymbol{G}) \subseteq \boldsymbol{L}(\boldsymbol{M})$.

Для того чтобы доказать обратное включение $\boldsymbol{L}(\boldsymbol{M})\subseteq\boldsymbol{L}(\boldsymbol{G})$ и, следовательно, равенство $\boldsymbol{L}(\boldsymbol{G})=\boldsymbol{L}(\boldsymbol{M})$, докажем, что

$$(q, xy, \$) \vdash^{n} (q, y, \$\eta A) \text{ влечет } \eta Ay \Rightarrow^{*} xy \qquad \text{(III.3.1.2)}$$

Утверждение этого доказывается индукцией по n.

Базис индукции. При $n=0$, выполняется автоматически.

Индукционный шаг. Предположим, что (III.3.1.2) верно для всех $n<m$. Если верхний символ стека автомата $\boldsymbol{M}$ нетерминальный, то последний такт был сделан по правилу (2) определения функции перехода δ. Поэтому можно писать

$$(q, xy, \$) \vdash^{m-1} (q, y, \$\eta\beta) \vdash (q, y, \$\eta A),$$

где $A\to\beta$ принадлежит $\boldsymbol{P}$. Если цепочка $\eta\beta$ содержит нетерминал, то по предположению индукции $\eta\beta y\Rightarrow^{*}xy$. Отсюда $\eta A y\Rightarrow\eta\beta y\Rightarrow^{*}xy$, что и утверждалось.

В качестве частного случая утверждения (III.3.1.2) получаем, что $(q, \omega, \$) \vdash^{*} (q, \varepsilon, \$S)$ влечет $S\Rightarrow^{*}\omega$. Так как СА $\boldsymbol{M}$ распознает цепочку ω только тогда, когда $(q, \omega, \$) \vdash^{*} (q, \varepsilon, \$S) \vdash (p, \varepsilon, \varepsilon)$, то отсюда следует, что $\boldsymbol{L}(\boldsymbol{M})\subseteq\boldsymbol{L}(\boldsymbol{G})$. Таким образом, $\boldsymbol{L}(\boldsymbol{M})=\boldsymbol{L}(\boldsymbol{G})$.

Примеры III.3.1.2. Построим восходящий анализатор $\boldsymbol{A}$ для грамматики $\boldsymbol{G}_0$. Пусть расширенный стековый автомат $\boldsymbol{A} = <\{q, r\}, T, q, \{r\}, \$, H, \vdash, \dashv, \perp, \delta>$, где δ определяется так:

(1) $\delta(q, b, \varepsilon) = \{(q, b)\}$ для всех $b\in\{T, +, *, (,)\}$;

(2) $\delta(q, \varepsilon, T*F) = \{(q, E)\}$;
$\delta(q, \varepsilon, T) = \{(q, E)\}$;
$\delta(q, \varepsilon, T*F) = \{(q, T)\}$;
$\delta(q, \varepsilon, F) = \{(q, T)\}$;
$\delta(q, \varepsilon, (E)) = \{(q, F)\}$;
$\delta(q, \varepsilon, a) = \{(q, F)\}$;

(3) $\delta(q, \varepsilon, \$E) = \{(r, \varepsilon)\}$.

Для входа $a+T*T$ распознаватель $\boldsymbol{A}$ может сделать следующую последовательность тактов:

$(q, a +T*T, \$) \vdash$

$(q, +T*T, \$a) \models$
$(q, +T*T, \$F) \models$
$(q, +T*T, \$T) \models$
$(q, +T*T, \$E) \models$
$(q, +T*T, \$E^+) \models$
$(q, +T*T, \$E^+a) \models$
$(q, +T*T, \$E^+F) \models$
$(q, +T*T, \$E^+T) \models$
$(q, +T*T, \$E^+T*) \models$
$(q, +T*T, \$E^+T*a) \models$
$(q, +T*T, \$E^+T*F) \models$
$(q, +T*T, \$E^+T) \models$
$(q, +T*T, \$E) \models$
$(q, \varepsilon, \varepsilon)$.

Заметим, что для входа T^+T*T распознаватель $\boldsymbol{A}$ может сделать много различных последовательностей тактов. Однако выписанная последовательность – единственная, которая переводит начальную конфигурацию в заключительную.

Задания III.3.1. Постройте СА для следующих БГ:

1. $\boldsymbol{G} = <\{x, a, p\}, \{A, D\}, \boldsymbol{P}, \{A\}>$, $\boldsymbol{P} = \{A \to x|Dx, D \to Da|p\}$
2. $\boldsymbol{G} = <\{a, b\}, \{A,B,C\}, \boldsymbol{P}, \{C\}>$, $\boldsymbol{P} = \{A \to b|Cb, B \to Aa|b, C \to Bb|a\}$.
3. $\boldsymbol{G} = <\{x,y,z\}, \{X,Y,Z,S\}, \boldsymbol{P}, S>$, $\boldsymbol{P} = \{S \to Xx|Yy, X \to Zz|z, Y \to Ba|b, Z \to y\}$

Вопросы III.3.1:

1. Что такое ε-правило?
2. Как определяется нормальная форма Грейбаха?
3. В какой форме должна быть БГ, чтобы построить эквивалентный ей СА?
4. Совпадает ли класс языков, порождаемых БГ с классом языков, распознаваемых СА?
5. Для заданной БГ $\boldsymbol{G}$ существует ли расширенный СА $\boldsymbol{M}$ такой, что $\boldsymbol{L(M)} = \boldsymbol{L(G)}$?

III.3.2. Алгоритмические проблемы бесконтекстных грамматик

В этом параграфе будут обсуждены алгоритмические проблемы бесконтекстных языков, а также будут предложены примеры, даны задания, сформулированы вопросы [1-13, 15-21, 25, 27-34].

1. Алгоритмически разрешимые проблемы.

Для бесконтекстных языков проблема пустоты языка и проблема бесконечности алгоритмически разрешимы.

Поскольку все способы описания бесконтекстных языков (бесконтекстные грамматики, стековые автоматы и др.) эквивалентны между собой, то решения перечисленных алгоритмических проблем достаточно показать только для одного из таких способов.

Разрешимость проблемы пустоты непосредственно следует из наличия алгоритмов приведения бесконтекстных грамматик, а именно: язык $\boldsymbol{L}(\boldsymbol{G})$ непуст, тогда и только тогда, когда в грамматике $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ начальный нетерминал S будет производящим, т.е. существует правило вида $S \rightarrow \tau$, $\tau \in \boldsymbol{T}^+$.

Проблема эквивалентности детерминированных стековых автоматов разрешима, т.е. справедлива следующая теорема.

Теорема III.3.2.1. Существует алгоритм, позволяющий по произвольным двум детерминированным стековым автоматам $\boldsymbol{M_1}$ и $\boldsymbol{M_2}$ узнать, верно ли, что $\boldsymbol{L}(\boldsymbol{M_1}) = \boldsymbol{L}(\boldsymbol{M_2})$.

Проблема бесконечности бесконтекстных грамматик разрешима, т.е. справедлива следующая теорема.

Теорема III.3.2.2. Существует алгоритм, позволяющий по произвольной бесконтекстной грамматике $\boldsymbol{G}$ узнать, является ли бесконечным язык $\boldsymbol{L}(\boldsymbol{G})$.

Свойства замкнутости класса бесконтекстных языков доказаны с помощью следующих теорем III.3.2.3, III.3.2.4, III.3.2.5:

Теорема III.3.2.3. Если $\boldsymbol{L}$— бесконтекстный язык, то $\boldsymbol{L}^*$ тоже бесконтекстный язык.

Доказательство. Пусть язык Lпорождается бесконтекстной грамматикой $G = <N, T, P, S>$. Тогда язык L^* порождается грамматикой $G_i = <N \cup \{R\}, T, P \cup \{R \to SR, R \to \varepsilon\}, S>$, где $S \notin N \cup T$.

Теорема III.3.2.4. Если L_1 *и* L_2 — бесконтекстные языки над алфавитом T, то $L_1 \cdot L_2$ тоже бессконтекстный язык.

Доказательство. Пусть язык L_1 порождается грамматикой $G_1 = <N_1, T, P_1, S_1>$, а язык L_2 порождается грамматикой $G_2 = <N_2, T, P_2, S_2>$, где $N_1 \cap N_2 = \emptyset$. Тогда $L_1 \cdot L_2$ порождается грамматикой $G_k = <N_1 \cup N_2 \cup \{R\}, T, P_1 \cup P_2 \cup \{R \to S_1 S_2\}, R>$, где $R \notin N_1 \cup N_2 \cup T$.

Теорема III.3.2.5. Если L_1 и L_2 — бесконтекстные языки над алфавитом T, то $L_1 \cup L_2$ тоже бессконтекстный язык.

Доказательство. Пусть язык L_1 порождается грамматикой $G_1 = <N_1, T, P_1, S_1>$, а язык L_2 порождается грамматикой $G_2 = <N_2, T, P_2, S_2>$, где $N_1 \cap N_2 = \emptyset$. Тогда $L_1 \cup L_2$ порождается грамматикой $G_k = <N_1 \cup N_2 \cup \{R\}, T, P_1 \cup P_2 \cup \{R \to S_1, R \to S_2\}, R>$, где $R \notin N_1 \cup N_2 \cup T$.

2. Алгоритмически неразрешимые проблемы

Существуют несколько алгоритмически неразрешимых проблем для бесконтекстных грамматик:

1. *Проблема однозначности*: существование алгоритма, который позволяет узнать однозначность БГ.
2. *Проблема замкнутости*: существование алгоритма, который позволяет узнать замкнутость языка относительно пересечения (дополнения), означающая, что пересечение двух бесконтекстных языков (дополнение к бесконтекстному языку) является бесконтекстным языком.
3. *Проблема пустоты дополнения* (*тривиальности*): существование алгоритма, который позволяет узнать пустоту дополнения языка, т.е. $T^* \backslash L(G) = \emptyset$?
4. *Проблема пустоты пересечения*: существование алгоритма, который позволяет узнать пуст ли пересечение двух языков,

порожденных двумя БГ $\boldsymbol{G_1}$ и $\boldsymbol{G_2}$, т.е. $\boldsymbol{L}(\boldsymbol{G_1}) \cap \boldsymbol{L}(\boldsymbol{G_2}) = \emptyset$?

5. *Проблема эквивалентности*: существование алгоритма, который позволяет узнать порождают ли две БГ $\boldsymbol{G_1}$ и $\boldsymbol{G_2}$ один и тот же язык, т.е. $\boldsymbol{L}(\boldsymbol{G_1}) = \boldsymbol{L}(\boldsymbol{G_2})$?

Задания III.3.2:

1. Найти бесконтекстную грамматику для языка $\boldsymbol{L_1} \cup \boldsymbol{L_2}$ где $\boldsymbol{L_1}$ порождается грамматикой $A \to a$, $A \to bA$, $A \to cAA$, а язык $\boldsymbol{L_2}$ — грамматикой $A \to aB$, $A \to cB$, $B \to aA$, $B \to bA$, $B \to \varepsilon$.
2. Для произвольной бесконтекстной грамматики $\boldsymbol{G}$ построить алгоритм, определяющий, пуст ли язык $\boldsymbol{L}(\boldsymbol{G})$.
3. Опишите бесконтекстный язык, порождаемый правилами $S \to AB$, $A \to Aa|bB$, $B \to a|Sb$.
4. Построить стековый автомат, эквивалентный бесконтекстной грамматике $\boldsymbol{N}=\{A,B,C\}$, $S=A$, $\boldsymbol{T}=\{a,b,c\}$, $\boldsymbol{P} = \{A \to aB,\ B \to aB,\ B \to b,\ B \to bD,\ D \to d,\ D \to dD,\ A \to aD,\ A \to a\}>$.
5. Для бесконтекстной грамматики $\boldsymbol{G} = \{\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S\}$ напишите алгоритм построения множества $\boldsymbol{L}=\{A \in \boldsymbol{N} : A \Rightarrow \varepsilon\}$.
6. Найти стековый автомат, распознающий язык, порождаемый грамматикой $S \to aSb$, $S \to bSa$, $S \to c$.

Найти стековый автомат, распознающий язык, порождаемый грамматикой $S \to 0S1$, $S \to 0F0$, $S \to 0$.

Вопросы III.3.2:

1. Какому типу грамматики относятся следующие правила $S \to T|T+S|T-S$, $T \to F|F*T$, $F \to a|b$?
2. Какому типу грамматики относятся следующие правила $S \to A+B|B+A$, $A \to a$, $B \to b$?
3. Можно ли построить стековый автомат, эквивалентный грамматике, заданной правилами $S \to AB$, $A \to Aa|bB$, $B \to a|\ Sb$?
4. Есть ли стековый автомат, эквивалентный заданной грамматике $\boldsymbol{G} = <\{A, B\}, \{0,1\}, \{X \to 0A|1S|\varepsilon\}, \{A \to 0B|1A\}, \{B \to 0S|1B\}, S>$?
5. Можно ли построить стековый автомат, эквивалентный грамматике с правилами $E \to E+E|E*E|(E)|i$?

IV. КОНТЕКСТНЫЕ ЯЗЫКИ

IV.1. Порождающие механизмы контекстных языков

IV.1.1. Контекстные грамматики

В этой части будут рассмотрены контекстные языки, обсуждены механизмы их порождения и распознавания, показаны их эквивалентность, а также будут предложены примеры, даны задания, сформулированы вопросы [1-4,6-9,11-13,15-21,24-32].

Механизмы порождения контекстных языков являются грамматики типа 1 и называются контекстно-зависимыми (контекстными) грамматиками. К ним относятся, грамматики непосредственно составляющих и неукорачивающие грамматики. Распознающим механизмом является линейно-ограниченные автоматы.

Определение IV.1.1.1. Контекстная грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ является грамматикой непосредственно составляющих, если каждое ее правило вывода имеет вид $\eta A\theta \rightarrow \eta\alpha\theta \in \boldsymbol{P}$, где $A \in \boldsymbol{N}$; $\eta, \theta \in (\boldsymbol{T} \cup \boldsymbol{N})^*$; $\alpha \in (\boldsymbol{T} \cup \boldsymbol{N})^+$.

Примеры IV.1.1.1. Пусть задана грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, $\boldsymbol{N} = \{S, A, B,\}$, $\boldsymbol{T} = \{a, b\}$ и множество правил $\boldsymbol{P}$ состоит из:

$$S \rightarrow ASBA$$
$$S \rightarrow AbA$$
$$A \rightarrow a$$
$$bB \rightarrow bb$$
$$AB \rightarrow BA$$

Эта грамматика $\boldsymbol{G}$ является грамматикой непосредственно составляющих.

Определение IV.1.1.2. Контекстная грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ является *неукорачивающей* (*noncontracting*), если всем ее правилам

вывода вида $\alpha \to \beta \in \boldsymbol{P}$ накладывается ограничение $|\alpha| \le |\beta|$, где α, $\beta \in (\boldsymbol{T} \cup \boldsymbol{N})^+$.

Примеры IV.1.1.2. Пусть задана грамматика с правилами:

$S \to TS$,	$A \to a$,
$S \to US$,	$TA \to AAT$,
$S \to b$,	$UAb \to b$,
$Tb \to Ab$,	$UAAA \to AAU$,

Эта грамматика не является неукорачивающей, так как последние три правила не имеют требуемого вида.

Пусть задана контекстная грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, тогда:

1) Если $\boldsymbol{G}$ не содержит правило с пустой правой частью, т.е. правило вида $A \to \varepsilon$, то она является неукорачивающей грамматикой.

2) Если $\boldsymbol{G}$ содержит правило с пустой правой частью, т.е. правило вида $A \to \varepsilon$, то она является укорачивающей грамматикой. Здесь длина выводимой цепочки при переходе от k-го шага к $(k+1)$-му шагу уменьшается.

Определение IV.1.1.3. *Я*зыки, порождаемые контекстными грамматиками, называются контекстными языками.

Примеры IV.1.1.3. Пусть задана КГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где:

1. $\boldsymbol{P} = \{S \to aSBC | abc,\ cB \to BC,\ bB \to bb,\ bC \to bc,\ cC \to cc\}$, $\boldsymbol{N} = \{B, C, S\}$, $\boldsymbol{T} = \{a, b, c\}$. Тогда грамматика $\boldsymbol{G}$ порождает контекстный язык $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n b^n c^n : n \ge 1\}$ с помощью цепочкой вывода: $S \to aSBC \to aabcBC \to aabBCC \to aabbCC \to aabbcC \to aabbcc$.

2. $\boldsymbol{P} = \{A \to aAB,\ A \to Abc,\ bB \to bbc,\ cB \to Bc\}$, $\boldsymbol{N} = \{A, B, C\}$, $\boldsymbol{T} = \{a, b, c\}$. Тогда грамматика $\boldsymbol{G}$ порождает контекстный язык $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n b^n c^n : n \ge 1\}$ с помощью следующей цепочкой вывода $A \to aAB \to aabcB \to aabBc \to aabbcc$.

3. $\boldsymbol{P} = \{S \to SbaS,\ S \to ab,\ aS \to b\}$, $\boldsymbol{N} = \{S\}$, $\boldsymbol{T} = \{a, b\}$. Тогда грамматика $\boldsymbol{G}$ порождает контекстный язык $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{ab^{n+1}\}$ с помощью вывода $S \to SbaS \to abbaS \to abbb$, $S \to SbaS \to abbaab$.

4. $\boldsymbol{P}$ = {$S \to ABA$, $B \to ABCA$, $B \to b$, $bC \to bb$, $AC \to DC$, $DC \to DA$, $DA \to CA$, $A \to a$}, $\boldsymbol{N}$ = {A, B, C, D}, $\boldsymbol{T}$ = {a, b, c}. Тогда язык, порожденный этой грамматикой $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n b^n c^n : n \geq 1\}$ будет контекстным языком.

Покажем, что для произвольной контекстной грамматики, порождающей язык без пустой цепочки, можно построить эквивалентную неукорачивающую грамматику.

Пусть задана контекстная грамматика $\boldsymbol{G}$ = <$\boldsymbol{N}$, $\boldsymbol{T}$, $\boldsymbol{P}$, S>. Построим множество всех нетерминальных символов грамматики, из которых выводится пустая цепочка, выделив следующие множества:

$$W_1 = \{A: A \to \varepsilon \in \boldsymbol{P}\},$$

$$W_{n+1} = W_n \cup \{B: B \to \phi \in \boldsymbol{P}, \phi \in W_n^*\}$$

Затем найдем множество правил эквивалентной грамматики в два этапа:

а) удалив из множества $\boldsymbol{P}$ исходной грамматики правила с пустой правой частью $\boldsymbol{P}_1 = \boldsymbol{P} \setminus \{A \to \varepsilon : A \to \varepsilon \in \boldsymbol{P}\}$;

б) получив новые правила $A \to \phi'$ после удаления из каждого правила исходной грамматики $A \to \phi \in \boldsymbol{P}$ те нетерминалы, которые вошли в множество W_n по правилу:

$$\boldsymbol{P}_2 = \{A \to \phi' | A \to \phi \in \boldsymbol{P_1}, \phi = \phi_1 B \phi_2 | B \in W_n, \phi' = \phi_1 B \phi_2\}.$$

Повторив пункт б) для каждого нетерминала, принадлежащего множеству W_n, получим эквивалентную грамматику $\boldsymbol{G}$ = <$\boldsymbol{N}$, $\boldsymbol{T}$, $\boldsymbol{P_2}$, S>.

Таким образом, справедлива следующая теорема.

Теорема IV.1.1.1. Каждая контекстная грамматика является неукорачивающей грамматикой.

Примеры IV.1.1.4. Пусть задана грамматика $\boldsymbol{G}$ со следующими правилами вывода:

$$S \to AbA | cAb | Bb$$

$$A \to aAb | \varepsilon$$

$$B \to AA | a$$

Необходимо построить:

1) множество нетерминальных символов, из которых выводится пустая цепочка ε;

2) неукорачивающую грамматику, эквивалентную исходной грамматике.

Для того чтобы построить множество всех нетерминалов грамматики, из которых выводится пустая цепочка, выделим следующие множества:

$$W_1 = \{A:\ A \to \varepsilon \in \boldsymbol{P}\};$$

$$W_{m+1} = W_m \cup \{B:\ B \to \phi \in \boldsymbol{P},\ \phi \in W_m\ ^*\}.$$

Так как мы имеем правило $A \to \varepsilon \in \boldsymbol{P}$, то можно построить множество $W_1 = \{A\}$, включающее нетерминал A.

Построим множество W_2. С нетерминалом A связан нетерминал B, т.е. существует правило $B \to AA$ и $A \in W_1$. Следовательно, $W_2 = \{A, B\}$.

$W_3 = W_2$, т. к. множество $W_3 = \{B: B \to \phi \in \boldsymbol{P}, \phi \in W_m\ ^*\}$ является пустым.

Исключив правило, содержащее пустую цепочку в правой части, получим неукорачивающую грамматику $\boldsymbol{G}_1$ вида:

$S \to AbA|cAb|Bb|bA|Ab|cb|b,\ A \to aAb|ab,\ B \to AA|A|a.$

Определение IV.1.1.4. Контекстная грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$ находится в нормальной форме Куроды, если каждое ее правило вывода для A, A', B, $B' \in \boldsymbol{N}$, $a \in \boldsymbol{T}$ будет представимо в одной из следующих форм:

$A \to BB'$,

$AB \to A'B$,

$A \to a$,

$A \to AB'$,

$A \to A'$,

По любой заданной контекстной грамматике $\boldsymbol{G}_1$ можно построить эквивалентную ей контекстную грамматику $\boldsymbol{G}_2$ в нормальной форме Куроды такую, что их порождаемые языки будут совпадать, т.е. $\boldsymbol{L}(\boldsymbol{G}_2) = \boldsymbol{L}(\boldsymbol{G}_1)$.

Замечание IV.1.1.1. В данной грамматике может быть несколько эквивалентных выводов, в которых применяются одни и те же правила в одних и тех же местах, но в различном порядке. Для произвольных грамматик сложно определить эквивалентность двух выводов.

Примеры IV.1.1.5.

1**.** Пусть задана контекстная грамматика $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{P} = \{S \to aSBC\ ,\ abc,\ cB \to BC,\ bB \to bb,\ bC \to bc,\ cC \to cc\}$, $\boldsymbol{N} = \{B, C, S\}$, $\boldsymbol{T} = \{a, b, c\}$. Тогда грамматика $\boldsymbol{G}$ порождает контекстный язык $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n b^n c^n : \ n \geq 1\}$ с помощью следующих правил вывода: $S \to aSBC \to aabcBC \to aabBCC \to aabbCC \to aabbcC \to aabbcc$.

2. Пусть задана КГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N} = \{A, B, C\}$, $\boldsymbol{T} = \{a, b, c\}$, $\boldsymbol{P} = \{A \to aAB,\ A \to Abc,\ bB \to bbc,\ cB \to Bc\}$. Тогда грамматика $\boldsymbol{G}$ порождает контекстный язык $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n b^n c^n : \ n \geq 1\}$ с помощью следующего вывода: $A \to aAB \to aabcB \to aabBc \to aabbcc$.

3. Пусть задана КГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N} = \{S\}$, $\boldsymbol{T} = \{a, b\}$, $\boldsymbol{P} = \{S \to SbaS,\ S \to ab,\ aS \to b\}$. Тогда грамматика $\boldsymbol{G}$ порождает контекстный язык $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{ab^{n+1}\}$ с помощью следующего вывода:

$$S \to SbaS \to abbaS \to abbb,\ S \to SbaS \to abbaab.$$

Задания IV.1.1:

1. Описать язык, порождаемый грамматикой с правилами:

$S \to A1$
$A \to 1A1 | 0A0 | B$
$B \to 021 | 120C$
$C1 \to 0C$
$C0 \to 1$
$R1 \to 11$

2. Найти неукорачивающую грамматику, порождающую язык:

$\{\omega \in \{a,b,c,\}^+ : \ |\omega|_a = |\omega|_b = |\omega|_c\}$;
$\{\omega \in \{a,b,c,\}^* : \ |\omega|_a < |\omega|_b < |\omega|_c\}$;
$\{\omega c \omega : \ \omega \in \{a,b,c,\}^*\}$;
$\{c^n d c^n d c^n : \ n \geq 0\}$;

$\{a^n b^n c^n : n \geq 1\}$.

3. Доказать, что грамматика с правилами:

$S \rightarrow aSBC | abC$

$CB \rightarrow BC$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

порождает язык $\boldsymbol{L} = \{a^n b^n c^n : n \geq 1\}$.

4. Определить тип языка, порождаемого грамматикой:

$S \rightarrow 0A1,\ 0A \rightarrow 00A1,\ A \rightarrow \varepsilon$.

5. Построить КГ, порождающая язык $\boldsymbol{L} = \{\omega : \omega \in \{a, b, c\}$ и в цепочке ω число букв a равно числу букв b *и* равно числу букв $c\}$.

Вопросы IV.1.1:

1. К какому типу относится грамматика с правилами?

a) $S \rightarrow a | Ba,\ B \rightarrow Bb | b$

b) $S \rightarrow Ab,\ A \rightarrow Aa | ba$

c) $S \rightarrow 0A1 | 01,\ 0A \rightarrow 00A1,\ A \rightarrow 01$

d) $S \rightarrow AB,\ AB \rightarrow BA,\ A \rightarrow a,\ B \rightarrow b$

e) $S \rightarrow aAb,\ aA \rightarrow aaAb,\ A \rightarrow \varepsilon$

f) $S \rightarrow AB | ABS,\ AB \rightarrow BA,\ BA \rightarrow AB,\ A \rightarrow a,\ B \rightarrow b$

h) $S \rightarrow abC | aB,\ B \rightarrow bc,\ bC \rightarrow bc$

2. Пусть задана грамматика с единственным нетерминалом K, терминалами 1, 2, 3 и правилами $K \rightarrow 0$, $K \rightarrow K1$, $K \rightarrow KK2$, $K \rightarrow KKK3$.

Как проверить выводимость слова в этой грамматике, читая слово слева направо (число действий при прочтении одной буквы должно быть ограничено)?

3. Какому типу относится язык:

$\boldsymbol{L} = \{1^{3n+2} 0^n : n \geq 0\}$?

$\boldsymbol{L} = \{a^m b^n a^m b^n : m, n \geq 1\}$;

$\boldsymbol{L} = \{\omega\omega : \omega \in \{a, b\}\}$;

$\boldsymbol{L} = \{0^i 1^j : i \neq j;\ i, j \geq 0\}$;

$\boldsymbol{L} = \{a^n b^n c^n : n \geq 1\}$;

$\boldsymbol{L} = \{ ab^{n+1} \}.$

IV.2. Распознающие механизмы контекстных языков

IV.2.1. Линейно-ограниченные автоматы

В этой части будут описаны состав, структура и функции линейно-ограниченных автоматов (ЛОА), дается его формальное определение, конфигурация, такт и способ распознавания языка, а также будут предложены примеры, даны задания, сформулированы вопросы [1-4,6-9,11-13,15-21,24-32].

Линейно-ограниченные автоматы (*linear-bounded automaton*) похожи на недетерминированные конечные автоматы, но их головки могут записывать символы на входную ленту и двигаться в обе стороны - на одну ячейку вправо или на одну ячейку влево. На рисунке IV.2.1 показана структура линейно-ограниченного автомата.

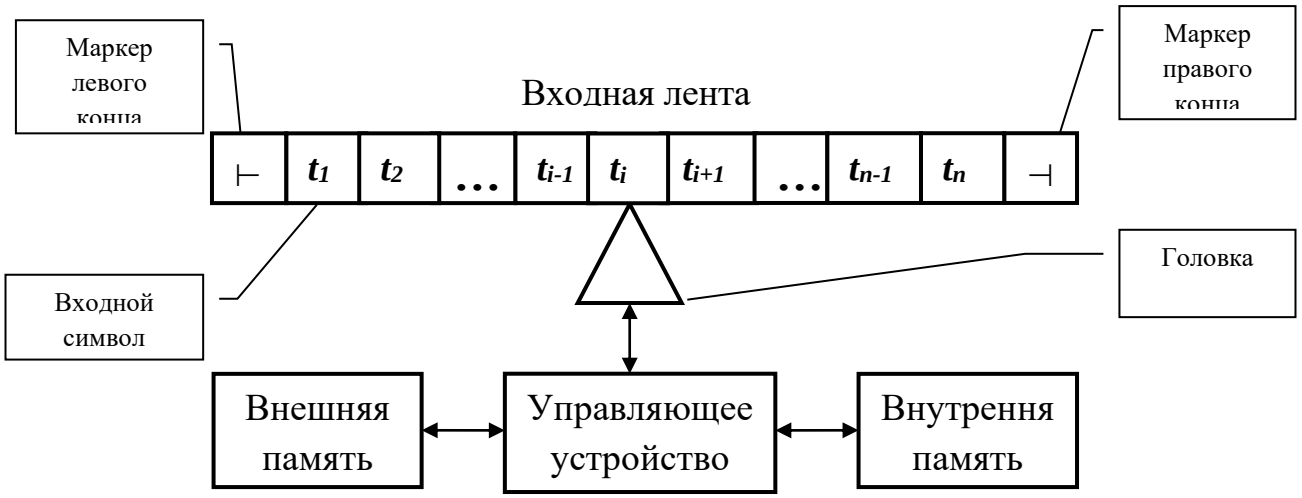


Рисунок IV.2.1. Структура линейно-ограниченного автомата

Определение IV.2.1. Недетерминированным линейно-ограниченным автоматом (НЛОВ) называется следующая десятка $\boldsymbol{M} = \langle \boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{U}, q_s, \boldsymbol{F}, \vdash, \dashv, \nleftarrow, \nrightarrow, \delta \rangle$, где:

$\boldsymbol{Q}$ – конечное множество состояний управляющего устройства;

$\boldsymbol{U}$ – конечное множество ленточных символов, $\boldsymbol{Q} \cap \boldsymbol{U} = \varnothing$;

$\boldsymbol{T}$ – конечное множество входных символов, $\boldsymbol{T} \subseteq \boldsymbol{U}$;

q_s – начальное состояние управляющего устройства, $q_s \in \boldsymbol{Q}$;

$\boldsymbol{F}$ – множество заключительных состояний управляющего устройства, $\boldsymbol{F} \subseteq \boldsymbol{Q}$;

$\vdash,\dashv$ – маркер начало, маркер конца ленты, $\vdash,\dashv \in \boldsymbol{U}$, $\vdash,\dashv \notin \boldsymbol{T}$;

$\nleftarrow$ – специальный символ, показывающий передвижение головки налево;

$\nrightarrow$ – специальный символ, показывающий передвижение головки направо;

δ: $\boldsymbol{Q}\times\boldsymbol{T}\to\boldsymbol{Q}\times\boldsymbol{T}\times\{\nleftarrow,\nrightarrow\}$ – функция перехода для ДЛОА.

δ: $\boldsymbol{Q}\times\boldsymbol{U}\to\vartheta(\boldsymbol{Q}\times\boldsymbol{U}\times\{\nleftarrow, \nrightarrow\})$ – функция перехода для НЛОА, где $\vartheta(\boldsymbol{Q}\times\boldsymbol{U}\times\{\nleftarrow, \nrightarrow\})$ – множество всех подмножеств множества $\boldsymbol{Q}\times\boldsymbol{U}\times\{\nleftarrow,\nrightarrow\}$.

Отсюда видно, что если для любого $q\in\boldsymbol{Q}$ и $t\in\boldsymbol{U}$ отображение $\delta(q, t)$ содержит не более одного элемента, то линейно ограниченный автомат будет *детерминированным*.

Определение IV.2.2. Конфигурацией линейно-ограниченного автомата называют тройку (q, χ, i), где:

(1) $q\in\boldsymbol{Q}$ – текущее состоянние управляющего устройства;

(2) $\chi = \vdash t_1 t_2 \ldots t_n \dashv$ – цепочка ленты, $t_j\in\boldsymbol{T}$, $1 \le j \le n$, причем, если $\chi = \vdash\varepsilon\dashv$, то считается, что лента полностью просмотрена;

(3) $i\in\boldsymbol{N}$ – натуральное число, задающее растояние головки от левого края цепочки τ.

Если $(p, t, \nleftarrow)\in\delta(q, t_i)$, $t\in\boldsymbol{T}$, $i>1$, тогда такт автомата $\boldsymbol{M}$ записывается с помощью следующего бинарного отношения:

$$(q, \vdash t_1 t_2 \ldots t_{i-1}\, t_i\, t_{i+1} \ldots t_n \dashv, i) \models_{\boldsymbol{M}} (p, \vdash t_1 t_2 \ldots t_{i-1} t\, t_{i+1} \ldots t_n \dashv, i\text{-}1)$$

Если $(p, t, \nrightarrow)\in\delta(q, t_i)$, $t\in\boldsymbol{T}$, $i < n$, тогда такт автомата $\boldsymbol{M}$ записывается с помощью следующего бинарного отношения:

$$(q, \vdash t_1 t_2 \ldots t_{i-1}\, t_i\, t_{i+1} \ldots t_n \dashv, i) \models_{\boldsymbol{M}} (p, \vdash t_1 t_2 \ldots t_{i-1}\, t\, t_{i+1} \ldots t_n \dashv, i+1)$$

Отношение $\models_{\boldsymbol{M}}$ показывает, что головка автомата $\boldsymbol{M}$ записав в i-тую ячейку символ t вместо t_i и изменив состояние q на состояние p, сдвигается на одну ячейку влево ($\nleftarrow$) или вправо ($\nrightarrow$) от нее, не выходя за пределы области, в которой входная цепочка находилась

изначально. При этом, если известен автомат, то букву $\boldsymbol{M}$ при отношении $\models_{\boldsymbol{M}}$ можно опустить.

Как обычно, через $\models^k$ записывается k-тая степень бинарного отношения $\models$, а ее рефлексивное и транзитивное замыкание обозначается через $\models^*$.

Язык $\boldsymbol{L}(\boldsymbol{M})$, распознаваемый ЛОА $\boldsymbol{M}$ определяется так:

$$\boldsymbol{L}(\boldsymbol{M}) \rightleftharpoons \{\tau:\ \tau \in \boldsymbol{T}^* \& (q_s, \vdash\tau\dashv, 1) \models^* (q, \vdash\alpha\dashv, i) \& q \in \boldsymbol{F} \& \alpha \in \boldsymbol{T}^* \& 1 \le i \le n,\ n = |\tau| + 2\}$$

Вообще говоря, язык называется распознаваемым, если существует алгоритм, который за конечное число шагов позволяет получить ответ о принадлежности любой цепочки языку.

Замечание IV.2.1. Если нужно написать программу, моделью которой является линейно-ограниченный автомат, то нужно считать, что он будет недетерминированным автоматом.

Примеры IV.2.1. Построить линейно-ограниченный автомат (ЛОА), распознающий контекстного языка $\boldsymbol{L}(\boldsymbol{G}_1)=\{a^n b^n a^n\}$, если контекстная грамматика $\boldsymbol{G}_1$ имеет следующие правила вывода:

$S \to ACA$,
$C \to ACBA$,
$C \to b$,
$AB \to KB$,
$KB \to KA$,
$KA \to BA$,
$bB \to bb$,
$A \to a$.

Рассмотрим работу ЛОА при анализе терминальной цепочки *aaabbbaaa*, выводимой в данной грамматике следующим образом:

$S \to ACA \to AACBAA \to$
$AAACBABAA \to AAACBKBAA \to$
$AAACBKAAA \to AAACBBAAA \to$
$AAAbBBAAA \to AAAbbBAAA \to$
$AAAbbbAAA \to aAAbbbAAA \to$

$$aaAbbbAAA \rightarrow aaabbbAAA \rightarrow$$
$$aaabbbaAA \rightarrow aaabbbaaA \rightarrow aaabbbaaa.$$

Нужно учесть, что вывод цепочки $a^3b^3a^3 \in \boldsymbol{L}(\boldsymbol{G_1})$ возможен в эквивалентной грамматике со следующими правилами:

$$S \rightarrow aCa,$$
$$C \rightarrow aCBa,$$
$$C \rightarrow b,$$
$$aB \rightarrow Ba \; bB \rightarrow bb.$$

Представим конфигурации ЛОА, в которые он приходит при распознавании следующей цепочки:

$a_0q_1\ aaabbbaaa\ a_0 \vdash a_0D\ q_1aabbbaaa\ a_0C \vdash$
$a_0Daabbbaaaq_1 \vdash \ldots \vdash q_2Daabbbaaa A \vdash \ldots \vdash$
$Daabq_2bbaaA \vdash \ldots \vdash q_2DaabbBaaA \vdash \ldots \vdash$
$Daaq_2bbBaaA \vdash \ldots \vdash q_2DaabBBaaA \vdash \ldots \vdash$
$Daaq_2bBBaaA \vdash \ldots \vdash q_2DaaCBBaaA \vdash \ldots \vdash$
$DaaCBq_2BaaA \vdash \ldots \vdash q_2DaaCBaBaA \vdash \ldots \vdash$
$Daq_2aCBaBaA \vdash \ldots \vdash q_2DaCннBaA \vdash \ldots \vdash$
$Dq_2aCннBaA \vdash \ldots \vdash q_2Dcнннннн A \vdash q_2* \vdash$
$q_2aCнннннна \vdash \ldots \vdash q_2Sнннннн н \vdash \ldots \vdash Sнннннннн q_z$

Замечание IV.2.1. Если анализируемая цепочка не входит в язык $\boldsymbol{L}(\boldsymbol{G_1})=\{a^nb^na^n\}$, то ЛОА при любом варианте анализа перейдет в тупиковую кофигурацию. Например, один из вариантов анализа цепочки *abaa* следующий:

$q_1abaa \vdash \ldots \vdash q_2DbaA \vdash \ldots \vdash Dq_2baA \vdash \ldots \vdash q_2DcaA \vdash \ldots \vdash q_2aCaa$
$\vdash q_2Sнна \vdash \ldots \vdash Sннq_2a$ – тупик.

Итак, алгоритм распознавания входной цепочки инициальным ЛОА-ом $\boldsymbol{M}=<\boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{U}, q_s, \boldsymbol{F}, \vdash, \dashv, \nrightarrow, \rightarrow, \delta>$, где δ: $\boldsymbol{Q}\times\boldsymbol{U}\rightarrow\boldsymbol{Q}\times\boldsymbol{U}\times\{\nrightarrow,\rightarrow\}$ $\tau\in\boldsymbol{U}^*$, $|\tau| = \boldsymbol{L}$заключается в следующем:

– образуется всевозможные конфигурации автомата длиной $\boldsymbol{L}+1$ ($\boldsymbol{L}$- длина исходной цепочки $\tau\in\boldsymbol{U}^*$, к которой добавляется единица – символ состояния q_i). Таких конфигураций конечное множество;

– образуется всевозможные конфигурации ЛОА длиной $\boldsymbol{L}+1$ ($\boldsymbol{L}$-длина исходной цепочки $\tau\in\boldsymbol{U}^*$, к которой добавляется единица – символ состояния q_i). Таких конфигураций конечное множество;

– образуется из этих конфигураций всевозможные их последовательности без повторения (т.е. каждая конфигурация может входить в кортеж только один раз). Очевидно, что и такое множество наборов также конечно.

Выделяется подмножество последовательностей, начинающихся с конфигурации $(q_s, \vdash\tau\dashv, 1)$ до конфигурации

$$(q, \vdash\alpha\dashv, i)\&q\in\boldsymbol{F}\&\alpha\in\boldsymbol{T}^*\&1\le i\le n\&n=|\tau|+2).$$

Если среди этих последовательностей найдется такая, которая определяет возможный вариант работы ЛОА, цепочка $\tau\in\boldsymbol{U}^*$ распознается, в противном случае – не распознается.

Задания IV.2.1. Постройте линейно-ограниченные автоматы, распознающие следующие языки:

1. $\{a^nb^nc^n:\ n\ge1\}$;
2. $\{a^nb^nc^nd^n:\ n\ge1\}$;
3. $\{(ab)^n(ba)^n(ab)^n:\ n\ge1\}$;
4. $\{\tau c\tau:\ \tau\in\{a,b\}^*\}$;
5. $\{\omega\in\{a,b,c\}^*:\ |\omega|_a<|\omega|_b<|\omega|_c\}$;
6. $\{\omega\in\{a,b,c\}^*:\ \omega|_a=|\omega|_b=|\omega|_c\}$;
7. $\{0^m1^n: m\ne n\ \&\ m\ge0\ \&\ n\ge0\}$.

Вопросы IV.2.1. Является линейно-ограниченным автоматом распознаватель следующих языков:

1. $\{ab^{n+1}\}$?
2. $\{\tau c\tau:\ \tau\in\{a,b,c\}^*\}$?
3. $\{\tau\in\{a,b,c,d\}^*:\ |\tau|_a\ne|\tau|_b,\ |\tau|_c\ne|\tau|_d\}$?
4. $\{\tau\in\{a,b,c\}^*:\ |\tau|_a<|\tau|_b<|\tau|_c\}$?
5. $\{\omega\omega^R:\ \omega\in\{a,b\}\ \&\ \omega=ab,\ \omega^R=ba\}$?
6. $\{c^nd c^n d c^n: n\ge0\}$?
7. $\{a^n(ba)^nba^n: n\ge1\}$?

IV.3. Свойства контекстных языков

IV.3.1. Эквивалентность контекстных грамматик и линейно-ограниченных автоматов

В этом разделе будут рассмотрены свойства контекстных языков, показывается эквивалентность контекстных грамматик и линейно-ограниченных автоматов, обсуждены алгоритмические проблемы контекстных языков, а также будут предложены примеры, даны задания, сформулированы вопросы [1-4,6-9,11-13,15-21,24-32].

Можно показать, что класс языков, порождаемых контекстными грамматиками (КГ), в точности совпадает с классом языков, распознаваемых линейно-ограниченными автоматами (ЛОА). Это доказывается построением недетерминированного ЛОА (НЛОА), соответствующего данной КГ, и наоборот, построением КС-грамматики, соответствующей даному ЛОА.

Лемма IV.3.1. Пусть язык $\boldsymbol{L}(\boldsymbol{M})$ порожден КГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, тогда существует НЛОА $\boldsymbol{M}$, такой, что $\boldsymbol{L}(\boldsymbol{M}) \subseteq \boldsymbol{L}(\boldsymbol{G})$.

Доказательство. Нам необходимо построить НЛОА $\boldsymbol{M}$, распознающий язык $\boldsymbol{L}(\boldsymbol{M})$. Не вдаваясь в детали построения ЛОА $\boldsymbol{M}$, поскольку он довольно сложен, рассмотрим схему его работы.

Допустим, что лента автомата имеет два трека. Первый трек будет содержать входную цепочку τ с концевыми маркерами, т.е. содержимое первого трека имеет вид $\vdash\tau\dashv$, где $\tau = t_1\ t_2\ \ldots\ t_n$ – входная цепочка, $t_i \in \boldsymbol{T}$, $1 \le i \le n$, а символы $\vdash \notin \boldsymbol{T}$ и $\dashv \notin \boldsymbol{T}$ – маркер начало и маркер конца ленты. Второй трек будет содержать вычисляемую цепочку $\upsilon = u_1\ u_2\ \ldots\ u_n$, $u_i \in \boldsymbol{N} \cup \boldsymbol{T} \cup \{\$\}$, $1 \le i \le n$, где \$ – символ заполнитель пустой ячейки, т.е. лента будет содержать пару: (τ, υ). В начальной конфигурации лента содержит цепочку

$$(\vdash,\$), (t_1, \$),\ldots, (t_n, \$), (\dashv,\$).$$

ЛОА $\boldsymbol{M}$ выполняет следующую процедуру генерации:

1. Начальный нетерминальный символ S помещается в самой левой ячейке второго трека;

2. Выбирается из второго трека подцепочка α такая, что правило $\alpha \to \beta \in \boldsymbol{P}$;

3. Подцепочка α заменяется на β;

4. Если $|\beta| > |\alpha|$, то во втором треке символы справа от α перемещаются вправо, при этом, если это может привести к перемещению символа за правый концевой маркер, то процедура генерации останавливается.

5. Недетерминированно выбирается перейти на шаг 2 или завершить работу процедуры генерации.

На выходе из процедуры генерации первый трек все еще содержит цепочку τ, а второй трек содержит цепочку υ такую, что $S \Rightarrow_{\boldsymbol{G}}^* \upsilon$. НЛОА $\boldsymbol{M}$ сравнивает символы первого трека с соответствующими символами второго трека. При этом:

1) если цепочки символов первого и второго треков не одинаковы, т.е. $\tau \neq \upsilon$, то считается, что НЛОА $\boldsymbol{M}$ останавливается без распознавания входной цепочки τ;

2) если цепочки символов первого и второго треков одинаковы, т.е. $\tau = \upsilon$, то НЛОА $\boldsymbol{M}$ останавливается и распознает цепочку τ.

Если $\tau \in \boldsymbol{L}(\boldsymbol{G})$, то существует некоторая последовательность шагов, на которой НЛОА $\boldsymbol{M}$ строит цепочку τ на втором треке и распознает вход. Аналогично, для того, чтобы НЛОА $\boldsymbol{M}$ распознал цепочку τ, должна существовать последовательность шагов такая, что τ может быть построена на втором треке. Таким образом, в $\boldsymbol{G}$ должен быть вывод цепочки τ из начального нетермнала S.

Лемма IV.3.2. Если язык $\boldsymbol{L}$ распознается НЛОА $\boldsymbol{M}$, т.е. $\boldsymbol{L} = \boldsymbol{L}(\boldsymbol{M})$, то существует КГ, порождающая этот язык $\boldsymbol{L}$, т.е. $\boldsymbol{L} = \boldsymbol{L}(\boldsymbol{G})$.

Доказательство. Заметим, что нетерминалы КГ должны указывать не только исходное и текущее содержимое ячеек ленты НЛОА, но и то, является ли ячейка соседней справа или слева с концевым маркером. Кроме того, в обозначении нетерминала состояние НЛОА должно также комбинироваться с символом под головкой, поскольку КГ не может иметь раздельные символы для концевых маркеров и состояния НЛОА, так как эти символы должны были бы заменяться на пустые

цепочки ε, когда цепочка превращается в терминальную, а ε-*порождения* в КГ запрещены.

Пусть задан НЛОА $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{T}, \boldsymbol{U}, q_s, \boldsymbol{F}, \vdash, \dashv, \nleftarrow, \nrightarrow, \delta>$, распознаваемый язык $\boldsymbol{L}$, причем $\varepsilon \notin \boldsymbol{L}$.

Теперь построим КГ $\boldsymbol{G}$, которая сначала недетерминированно порождает две копии представления некоторой цепочки из множества $\boldsymbol{T}^*$, затем моделирует действие НЛОА $\boldsymbol{M}$ на одной из этих копий. Если НЛОА $\boldsymbol{M}$ распознает цепочку, то грамматика $\boldsymbol{G}$ превращает вторую копию в терминальную цепочку. Если НЛОА $\boldsymbol{M}$ не распознает цепочку, то вывод в грамматике $\boldsymbol{G}$ никогда не приводит к терминальной цепочке. Положим $S=A_1$, $\boldsymbol{N} = \{A_1, A_2\} \cup \{[q, \vdash, x, t, \dashv], [\vdash, q, x, t, \dashv], [\vdash, x, t, q, \dashv], [q, x, t], [q, x, t, \dashv], [x, t, q, \dashv], [\vdash, x, t], [x, t], \quad [x, t, \dashv]: t \in \boldsymbol{T}, x \in \boldsymbol{U} \setminus \{\vdash, \dashv\}, q \in \boldsymbol{Q}\}$, $\boldsymbol{T} = \boldsymbol{T}$, $\boldsymbol{P}$ включает:

(1) Моделирование начальной конфигурации вида $(q_0, \vdash t \dashv, 1)$:

$$A_1 \rightarrow [q_0, \vdash, t, t, \dashv];$$

(2) Моделирование движения на односимвольной цепочке при $q \in \boldsymbol{Q} \setminus \boldsymbol{F}$:

$$[q, \vdash, x, t, \dashv] \rightarrow [\vdash, p, x, t, \dashv], \text{ если } (p, \vdash, \nrightarrow) \in \delta(q, \vdash);$$

$$[\vdash, q, x, t, \dashv] \rightarrow [p, \vdash, y, t, \dashv], \text{ если } (p, y, \nleftarrow) \in \delta(q, x);$$

$$[\vdash, q, x, t, \dashv] \rightarrow [\vdash, y, t, p, \dashv], \text{ если } (p, y, \nrightarrow) \in \delta(q, x);$$

$$[\vdash, x, t, q, \dashv] \rightarrow [\vdash, p, x, t, \dashv], \text{ если } (p, \dashv, \nleftarrow) \in \delta(q, \dashv);$$

(3) Восстановление входной односимвольной цепочки при $q \in \boldsymbol{F}$:

$$[q, \vdash, x, t, \dashv] \rightarrow t;$$

$$[\vdash, q, x, t, \dashv] \rightarrow t;$$

$$[\vdash, x, t, q, \dashv] \rightarrow t;$$

(4) Моделирование начальных конфигураций вида $(q_0, \vdash \omega \dashv, 1)$, $|\omega| > 1$:

$$A_1 \rightarrow [q_0, \vdash, t, t] A_2;$$

$$A_2 \rightarrow [t, t] A_2;$$

$$A_2 \rightarrow [t, t, \dashv];$$

(5) Моделирование движения на левом конце цепочки при $q \in \boldsymbol{Q} \setminus \boldsymbol{F}$:

$$[q, \vdash, x, t] \rightarrow [\vdash, p, x, t], \text{ если } (p, \vdash, \nrightarrow) \in \delta(q, \vdash);$$

$$[\vdash, q, x, t] \rightarrow [p, \vdash, y, t], \text{ если } (p, y, \nleftarrow) \in \delta(q, x);$$

$$[\vdash, q, x, t]\,[z, u] \rightarrow [\vdash, y, t]\,[p, z, u], \text{ если } (p, y, \nrightarrow) \in \delta(q, x);$$

(6) Моделирование движения в середине цепочки при $q \in \boldsymbol{Q} \setminus \boldsymbol{F}$:

$$[q, x, t]\,[z, u] \to [y, t]\,[p, z, u], \text{ если } (p, y, \twoheadrightarrow) \in \delta(q, x);$$

$$[z, u]\,[q, x, t] \to [p, z, u]\,[y, t], \text{ если } (p, y, \twoheadleftarrow) \in \delta(q, x);$$

$$[q, x, t]\,[z, u, \dashv] \to [y, t]\,[p, z, u, \dashv], \text{ если } (p, y, \twoheadrightarrow) \in \delta(q, x);$$

(7) Моделирование движения на правом конце цепочки при $q \in \boldsymbol{Q} \setminus \boldsymbol{F}$:

$$[q, x, t, \dashv] \to [y, t, p, \dashv], \text{ если } (p, y, \twoheadrightarrow) \in \delta(q, x);$$

$$[x, t, q, \dashv] \to [p, x, t, \dashv], \text{ если } (p, \dashv, \twoheadleftarrow) \in \delta(q, \dashv);$$

$$[z, u]\,[q, x, t, \dashv] \to [p, z, u]\,[y, t, \dashv], \text{ если } (p, y, \twoheadleftarrow) \in \delta(q, x);$$

(8) Восстановление входного символа при переходе в состояние $q \in \boldsymbol{F}$ на левом конце цепочки:

$$[q, \vdash, x, t] \to t;$$

$$[\vdash, q, x, t] \to t;$$

(9) Восстановление входного символа при переходе в состояние $q \in \boldsymbol{F}$ в середине цепочки.

$$[q, x, t] \to t;$$

(10) Восстановление входного символа при переходе в состояние $q \in \boldsymbol{F}$ на правом конце цепочки.

$$[q, x, t, \dashv] \to t;$$

$$[x, t, q, \dashv] \to t;$$

(11) Восстановление входной цепочки слева направо:

$$t[x, u] \to tu;$$

$$t[x, u, \dashv] \to tu;$$

(12) Восстановление входной цепочки справа налево:

$$[x, t]\,u \to tu;$$

$$[\vdash, x, t]\,u \to tu;$$

Здесь $p \in \boldsymbol{Q}$; $u \in \boldsymbol{T}$; $y \in \boldsymbol{U} \setminus \{\vdash, \dashv\}$; $z \in \boldsymbol{U} \setminus \{\vdash, \dashv\}$.

Теперь индукцией по числу движений НЛОА $\boldsymbol{M}$ нетрудно доказать, что если непустая цепочка τ распознается НЛОА $\boldsymbol{M}$, то τ выводима в КГ $\boldsymbol{G}$. И наоборот, индукцией по длине вывода можно доказать, что если цепочка τ выводима в КГ $\boldsymbol{G}$, то τ распознается НЛОА $\boldsymbol{M}$. Таким образом, если $\tau \in \boldsymbol{L}(\boldsymbol{G})$, то $\tau \in \boldsymbol{L}(\boldsymbol{M})$ и наоборот, если $\tau \in \boldsymbol{L}(\boldsymbol{M})$, то $\tau \in \boldsymbol{L}(\boldsymbol{G})$. Из лемм *I*V.3.1 и *I*V.3.2 следует теорема.

Теорема IV.3.1. Язык ***L***, не содержащий пустой цепочки, порождается некоторой неукорачивающей грамматикой тогда и только тогда, когда существует НЛОА, распознающий язык ***L***.

Задания IV.3.1. Постройте линейно-ограниченный автомат, эквивалентного контекстной грамматике $G = <N, T, P, S>$, где:

1.$N=\{A, B, S\}$, $T=\{a, b, c\}$, $P=\{S\to aAB, A\to abc, bB\to bbc, cB\to Bc\}$.

2. $N=\{A, B, S\}$, $T=\{a, b, c\}$, $P=\{S\to AB, A\to a, A\to ac, B\to b, B\to cb\}$.

3. $N=\{ S\}$, $T=\{a, b\}$, $P=\{ S\to SbaS, S\to ab, aS\to b\}$.

4. $N=\{A, B, C, D, S\}$, $T=\{a, b, c\}$, $P=\{S\to CD, C\to aCA, C\to bCB, AD\to aD, BD\to bD, Aa\to aA, Ab\to bA, Ba\to aB, Bb\to bB, C\to c, D\to c\}$.

5. $N=\{A, B, C, S\}$, $T=\{0, 1, 2\}$, $P=\{S\to A1, A\to 1A1|0A0|B, B\to 021|120C, C1\to 0C, C0\to 1, \to 1\to 11\}$.

Вопросы IV.3.1. Является ли линейно-ограниченным автоматами распознаватели, распознающие языки, которые порождаются следующими грамматиками $G = <N, T, P, S>$, где

1. $N=\{B, S\}$, $T=\{a, b\}$, $P=\{S\to a|Ba, B\to Bb|b\}$;
2. $N=\{A, B, S\}$, $T=\{a, b\}$, $P=\{S\to Ab, A\to Aa|ba\}$;
3. $N=\{A, B, S\}$, $T=\{a, b, c\}$, $P=\{S\to aAb|ab, aA\to aaAb, A\to ab\}$;
4. $N=\{A, B, S\}$, $T=\{a, b\}$, $P=\{S\to AB, AB\to BA, A\to a, B\to b\}$;
5. $N=\{A, B, S\}$, $T=\{a,b\}$, $P=\{S\to AB|ABS, AB\to BA, BA\to AB, A\to a, B\to b\}$;
6. $N=\{B, C, S\}$, $T=\{a, b, c\}$, $P=\{S\to abC|aB, B\to bc, bC\to bc\}$;
7. $N=\{A,S\}$, $T=\{0, 1\}$, $P=\{ S\to 0A1, 0A\to 00A1, A\to\varepsilon\}$?
8. $\{a^n b^m c^{n+m}: n \ge 1, m\ge 1\}$?
9. $\{a^n b^m c^{2n+3m}: n\ge 1, m\ge 1\}$?
10. $\{a^n b^m c^{nm}: n\ge 1, m\ge 1\}$?
11. $\{a^n b^m c^k: n\ge 1, m\ge 1, k\ge 1\}$?
12. $\{a(cd)^n a: n\ge 0\} \cup \{b(cd)^n b: n\ge 0\}$?

IV.3.2. Алгоритмические проблемы контекстных языков

В этом параграфе будут обсуждены алгоритмические проблемы контекстных языков, а также будут предложены примеры, даны задания, сформулированы вопросы [1-4,8,9,11-13,15-21,24-32].

Среди них до сих пор имеются открытые проблемы, т.е. до сих пор нет ответа об их алгоритмически разрешимости или неразрешимости. Нас больше интересуют алгоритмически разрешимые проблемы. Знание этих проблем очень важно при решении практических задач. Ниже перечисляются некоторые такие проблемы:

Проблема принадлежности в неукорачивающей грамматике алгоритмически разрешима, т.е. справедлива следующая теорема.

Теорема IV.3.2.1. Существует алгоритм, позволяющий по произвольной неукорачивающей грамматике G и по слову τ узнать, верно ли, что $\tau \in L(G)$.

Проблема замкнутости контекстных языков относительно операции принадлежности объединения, пересечения и дополнения алгоритмически разрешима, т.е. справедлива следующая теорема.

Теорема IV.3.2.2. Класс языков, порождаемых не укорачивающими грамматиками, то есть класс контекстных языков, замкнут относительно операций объединения, пересечения и дополнения.

Заметим, что замкнутость относительно операции дополнения доказали в 1987 году (независимо друг от друга) Нил Иммерман (Nei*LI*mmerma*n*) и Роберт Селепчени (Robert Szelepcs). Замкнутость относительно операции объединения очевидна, а пересечение выражается через объединение и дополнение.

Проблема пустоты в бесконтекстной грамматике алгоритмически разрешима, т.е. справедлива следующая теорема.

Теорема IV.3.2.3. Существует алгоритм, позволяющий по произвольной бесконтекстной грамматике G узнать, верно ли, что

$\boldsymbol{L}(\boldsymbol{G}) = \emptyset$.

Доказательство. Удалим из $\boldsymbol{G}$ все бесполезные символы, кроме начального символа. Если полученная грамматика содержит хотя бы одно правило, то $\boldsymbol{L}(\boldsymbol{G}) \neq \emptyset$.

Примеры IV.3.1:

1. Контекстный язык $\boldsymbol{L(G)} = \{a^n b^n c^n : n \geq 1\}$ порождается контекстной грамматикой $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N}=\{B, S\}$, $\boldsymbol{T}=\{a, b, c\}$, $\boldsymbol{P}=\{S \to aSB, S \to abc, bB \to bbc, cB \to Bc\}$, так как можно построить следующий вывод: $S \to aSB \to aabcB \to aabBc \to aabbcc$.
2. Контекстный язык $\boldsymbol{L(G)} = \{a^n b^n c^n : n \geq 1\}$ порождается контекстной грамматикой $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N}=\{B, C, S\}$, $\boldsymbol{T}=\{a, b, c\}$, $\boldsymbol{P}=\{S \to aSBC | abC, CB \to BC, bB \to bb, bC \to bc, cC \to cc\}$, так как можно построить следующий вывод:

$$S \to aSBC \to aabCBC \to aabBCC \to aabbCC \to aabbcC \to aabbcc.$$

3. Контекстный язык $\boldsymbol{L(G)} = \{(ab)^n : n \geq 1\}$ порождается контекстной грамматикой $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N}=\{A, B, C, D, S\}$, $\boldsymbol{T}=\{a, b\}$, $\boldsymbol{P}=\{S \to CD, C \to aCA, C \to bCB, AD \to aD, BD \to bD, Aa \to aA, Ab \to bA, Ba \to aB, Bb \to bB, C \to \varepsilon, D \to \varepsilon\}$, так как можно построить вывод: $abCBaD \to abCaBD \to abCabD \to ab\varepsilon abD \to ab\varepsilon ab\varepsilon \to abab$.
4. Контекстный язык $\boldsymbol{L(G)} = \{ab^{n+1} : n \geq 1\}$ порождается контекстной грамматикой $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N}=\{S\}$, $\boldsymbol{T}=\{a, b\}$, $\boldsymbol{P}=\{S \to SbaS, S \to ab, aS \to b\}$, так как можно построить следующий вывод $S \to SbaS \to abbaS \to abbb$ или $S \to SbaS \to abbaab$.
5. Контекстный язык $\boldsymbol{L(G)} = \{(abc)^n : n \geq 1\}$ порождается контекстной грамматикой $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N}=\{B,C,S\}$, $\boldsymbol{T}=\{a, b,c\}$, $\boldsymbol{P}=\{S \to abC | aB, B \to bc, bC \to bcS\}$, так как можно построить следующие выводы: $S \to abC \to abcS \to abcaB \to abcabc$ или $S \to aB \to abc$.
6. Контекстный язык $\boldsymbol{L(G)} = \{(01)^n : n \geq 1\}$ порождается контекстной грамматикой $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N}=\{A,B,C\}$, $\boldsymbol{T}=\{a,b,c\}$, $S=A$, $\boldsymbol{P}=\{A \to 01BA, A \to 01C, 1B \to 101, C \to \varepsilon\}$ так как можно построить следующий вывод: $A \to 01BA \to 01B01C \to 010101\varepsilon \to 010101$.

Задания IV.3.1:

1. Доказать, что цепочка, порожденная контекстной грамматикой $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, S>$, где $\boldsymbol{N} = \{A, B, C\}$, $\boldsymbol{T} = \{a, b, c\}$, $\boldsymbol{P} = \{A \to aAB, A \to Abc, bB \to bbc, cB \to Bc\}$, принадлежит языку $\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n b^n c^n : n \geq 1\}$.
2. Найти контекстную грамматику $\boldsymbol{G}$, порождающую язык
$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{\omega \in \{a, b, c\}^+ : |\omega|_a = |\omega|_b = |\omega|_c\}.$$
3. Найти контекстную грамматику $\boldsymbol{G}$, порождающую язык
$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n b^n a^n : n \geq 1\}.$$
4. Найти контекстную грамматику, порождающую язык
$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{c^n d c^n d c^n : n \geq 0\}.$$
5. Найти контекстную грамматику $\boldsymbol{G}$, порождающую язык
$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{(ab)^n (ba)^n (ab)^n : n \geq 1\}.$$
6. Найти контекстную грамматику $\boldsymbol{G}$, порождающую язык
$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^n (ba)^n b a^n : n \geq 1\}.$$
7. Найти контекстную грамматику $\boldsymbol{G}$, порождающую язык
$$\{a^n b^n c^n d^n : n \geq 1\}.$$
8. Найти контекстную грамматику $\boldsymbol{G}$, порождающую язык
$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{\omega c \omega : \omega \in \{a, b\}^*\}.$$
9. Найти контекстную грамматику $\boldsymbol{G}$, порождающую язык
$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{a^{2n} : n \geq 0\}.$$

Вопросы IV.3.2.1:

1. Является ли непустым язык, порождаемый КГ с правилами: $S \to aaC\boldsymbol{M}$, $S \to aaaKT$, $\boldsymbol{M} \to a\boldsymbol{M}b$, $\boldsymbol{M} \to b\boldsymbol{M}a$, $\boldsymbol{M} \to \varepsilon$, $C \to aCa$, $C \to bCb$, $K \to bB$, $K \to aB$, $B \to bKa$, $B \to ab$?
2. Грамматикой какого класса порождается следующий язык
$$\{a^m b^n a^m b^n : m \geq, n \geq 1\}?$$
3. Является ли бесконечным язык, порождаемый КГ с правилами:
$S \to aF$, $S \to aaCM$, $S \to aaaKF$, $F \to J$, $F \to a$, $J \to F$, $J \to b$, $M \to aMb$, $M \to bMa$, $M \to \varepsilon$, $C \to aCa$, $C \to bCb$, $K \to bT$, $K \to aT$, $T \to bKa$, $T \to ab$?
4. Грамматикой какого класса порождается следующий язык
$$\{0^i 1^j : i \neq j, i \geq 0, j \geq 0\}?$$
5. Грамматикой какого класса порождается следующий язык
$$\{\omega \in \{a, b, c, d\}^* : |\omega|_a \geq |\omega|_b, |\omega|_c \geq |\omega|_d\}?$$

V. НЕОГРАНИЧЕННЫЕ ЯЗЫКИ

V.1. Порождающие механизмы неограниченных языков

V.1.1. Неограниченные грамматики

В этом разделе будут рассмотрены неограниченные языки, порождаемые грамматиками типа 0 и распознаваемые машинами Тьюринга, показана эквивалентность неограниченных грамматик и машин Тьюринга, а также будут обсуждены алгоритмические вопросы неограниченных языков, предложены примеры, даны задания и поставлены вопросы [1-4,8,9,11-13,15-21,24-32].

В разделе I.3.1 дано формальное определение грамматики в виде следующей четверки $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$.

Напомним, что:

1. В грамматике типа 0 на правила вывода вида $\alpha \to \beta$ не накладываются ни какие ограничения, поэтому их называют грамматиками общего вида или неограниченными грамматиками;
2. Языком, порождаемым грамматикой $\boldsymbol{G}$ называется множество терминальных цепочек, которые выводятся из одного начального нетерминала S, т.е.

$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{\omega:\ \omega \in \boldsymbol{T}^*,\ S \Rightarrow^* \omega\}.$$

Примеры V.1.1:

1. Пусть в грамматике $\boldsymbol{G}$ с терминальными символами $\{a,\ b\}$, нетерминальными символами $\{S, A, B\}$ и правилами $S \to Sb$, $S \to SA$, $S \to SB$, $S \to \varepsilon$, $Aa \to aaA$, $Ab \to ab$, $Baaa \to aaB$, $Bab \to b$ выводима терминальная цепочка aab. Тогда в ней существует следующий вывод $S \Rightarrow^* BBTTTb \Rightarrow^* BBaaaaaaab \Rightarrow^* aab$.
2. Неограниченной грамматикой является четверка

$$\boldsymbol{G}_1 = <\{0, 1\}, \{A, S\}, \boldsymbol{P}, S>,$$

где $\boldsymbol{P} = \{S \to 0A1,\ 0A \to 00A1,\ A \to \varepsilon\}$.

Легко заметить, что грамматика $\boldsymbol{G}$ порождает цепочки, в которых число нулей и единиц равны, т.е. $\boldsymbol{L}(\boldsymbol{G}) = \{0^n 1^n:\ n>1\}$.

3. Неограниченный язык $\boldsymbol{L(G)} \rightleftharpoons \{a^2 b^{n^2-1}: \quad n \geq 1\}$ порождается неограниченной грамматикой типа 0 с правилами:

$$S \rightarrow aaCFD,\ F \rightarrow AFB|AB,$$
$$AB \rightarrow bBA,\ Ab \rightarrow bA,$$
$$AD \rightarrow D,\ Cb \rightarrow bC,$$
$$CB \rightarrow C,\ bCD \rightarrow \varepsilon$$

Задания V.1.1.

Для заданной неограниченной грамматики $\boldsymbol{G} = <\boldsymbol{T}, \boldsymbol{N}, \boldsymbol{P}, S>$ построить язык $\boldsymbol{L(G)}$, порождаемый этой грамматикой, где:

1. $\boldsymbol{T} = \{a,b,d\}$, $\boldsymbol{N} = \{A,B,D\}$, $S = A$, $\boldsymbol{P} = \{A \rightarrow aB,\ B \rightarrow aB,\ B \rightarrow b,\ B \rightarrow bD,\ D \rightarrow d,\ D \rightarrow dD,\ A \rightarrow aD,\ A \rightarrow a\}$.
2. $\boldsymbol{T} = \{a,b\}$, $\boldsymbol{N} = \{A,B,C,D,S\}$, $\boldsymbol{P} = \{S \rightarrow CD,\ C \rightarrow aCA,\ C \rightarrow bCB,\ AD \rightarrow aD,\ BD \rightarrow bD,\ Aa \rightarrow aA,\ Ab \rightarrow bA,\ Ba \rightarrow aB,\ Bb \rightarrow bB,\ C \rightarrow \varepsilon,\ D \rightarrow \varepsilon\}$.
3. $\boldsymbol{T} = \{a,b,c\}$, $\boldsymbol{N} = \{A,B,C,D,S\}$, $\boldsymbol{P} = \{S \rightarrow CD,\ C \rightarrow aCA,\ C \rightarrow bCB,\ AD \rightarrow aD,\ BD \rightarrow bD,\ Aa \rightarrow aA,\ Ab \rightarrow bA,\ Ba \rightarrow aB,\ Bb \rightarrow bB,\ C \rightarrow c,\ D \rightarrow c\}$.
4. $\boldsymbol{T} = \{a,b,c\}$, $\boldsymbol{N} = \{B,C,S\}$, $\boldsymbol{P} = \{S \rightarrow abC|aB,\ B \rightarrow bc,\ bC \rightarrow bc\}$.
5. $\boldsymbol{T} = \{a,b,c\}$, $\boldsymbol{N} = \{A,B,C,D,S\}$, $\boldsymbol{P} = \{S \rightarrow CD,\ C \rightarrow aCA,\ C \rightarrow bCB,\ AD \rightarrow aD,\ BD \rightarrow bD,\ Aa \rightarrow aA,\ Ab \rightarrow bA,\ Ba \rightarrow aB,\ Bb \rightarrow bB,\ C \rightarrow c,\ D \rightarrow c\}$.
6. $\boldsymbol{T} = \{a,b,c\}$, $\boldsymbol{N} = \{B,C,S\}$, $\boldsymbol{P} = \{S \rightarrow aSBC|abC,\ CB \rightarrow BC,\ bB \rightarrow bb,\ bC \rightarrow bc,\ cC \rightarrow cc\}$.
7. $\boldsymbol{T} = \{a,b\}$, $\boldsymbol{N} = \{A,B,S\}$, $\boldsymbol{P} = \{S \rightarrow AB,\ AB \rightarrow BA,\ A \rightarrow a,\ B \rightarrow b\}$.

Вопросы V.1.1. Можно ли построить неограниченную грамматику, которая порождает следующий язык:

1. $\{a^m b^n a^m b^n : \ m \geq 1 \ \& \ n \geq 1\}$?
2. $\{a^k b^k a^n b^n : \ k \geq 1 \ \& \ n \geq 1\}$?
3. $\{a^{k+n} b^{k-n}: \ k \geq 1 \ \& \ n \geq 1\}$?
4. $\{(a^k b^m) c^n : \ k \geq 1 \ \& \ m \geq 1 \ \& \ n \geq 1\}$?
5. $\{a^k b^m c^n : \ k \geq 1 \ \& \ m \geq 1 \ \& \ n \geq 1\}$?
6. $\{a^n b^m c^n : \ m \geq 1 \ \& \ n \geq 1\}$?
7. $\{a^{k-m} b^m c^{n-m}: k \geq 1 \ \& \ m \geq 1 \ \& \ n \geq 1\}$?

V.2. Распознающие механизмы неограниченных языков

V.2.1. Машины Тьюринга

В этом разделе рассматриваются машины Тьюринга. Описываются её состав, структура и функция. Дается формальное определение, конфигурации, такта и программы, а также языка, распознаваемого машины Тьюринга. Показываются примеры, предлагаются задания и задаются вопросы [1-4,8,9,11-13,15-21,24-32].

Машиной Тьюринга называется формализм, предложенный для уточнения понятия алгоритма, английским математиком Аланом Тьюрингом в 1936 году.

В состав машины Тьюринга входят ограниченная с левой стороны бесконечная *лента*; считывающая и пишущая *головка*; *устройства управления* и внутренняя и внешняя *память*.

Лента разбита на ячейки, в каждой из которых можно записать только один символ из конечного входного/выходного алфавита, символ для обозначения левого края ленты или символ заполнитель пустых ячеек

Головка в каждый момент времени обозревает только одну ячейку ленты и в зависимости от символа в этой ячейке и состояния управляющего устройства может изменить (записывать или стирать) содержимое ячейки и сдвигаться на одну ячейку влево или вправо, либо оставаться без движения на месте.

Управляющее устройство может находиться в одном из состояний, среди которых можно выделить начальное (стартовое) состояние и заключительное (финальное) состояние.

Машина Тьюринга в начальном состоянии начинает свою работу, а в заключительном состоянии останавливает свою работу.

Память состоит из двух частей: *внутренняя память*, содержит множество всевозможных состояний управляющего устройства, а *внешняя память* – множество всех входных и выходных символов.

Данными машины Тьюринга являются цепочки, которые образованы из символов входного и выходного алфавита, и содержатся на ленте.

Перед началом работы МТ управляюще устройство находится в начальном состоянии, в самой левой ячейке ленты будет записываться специальный символ $\vdash$, после нее первые n ячейки ленты содержат цепочку $t_1t_2\ldots t_n$ из символов входного и выходного в алфавите, а остальные ячейки заполняются специальным символом заполнителем ϕ. При этом специальные символы $\vdash$ и ϕ не являются элементами входного и выходного алфавитов.

Структура машины Тьюринга представлена на рисунке V.2.1

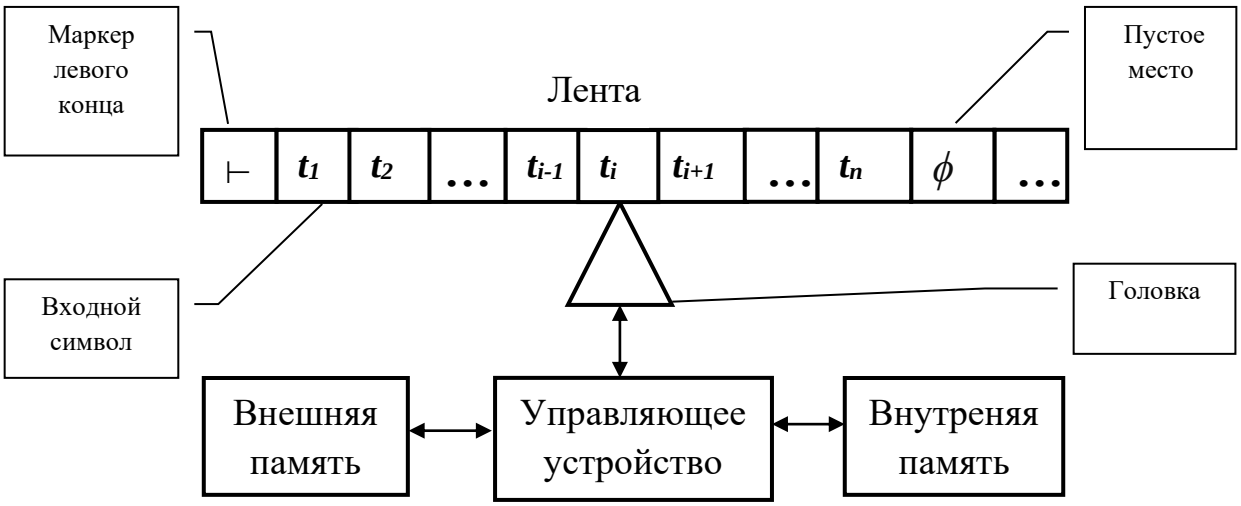


Рисунок V.2.1. Структура машины Тьюринга

Теперь дадим формальное (математическое) определение машины Тьюринга.

Определение V.2.1.1. Машиной Тьюринга (МТ) называют следующую десятку

$$\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{U}, \boldsymbol{T}, q_s, \boldsymbol{F}, \vdash, \Lsh, \Rsh, \|, \delta>, \text{ где}$$

$\boldsymbol{Q}$ – конечное множество состояний управляющего устройства;

$\boldsymbol{U}$ – конечное множество ленточных символов, $\boldsymbol{Q} \cap \boldsymbol{U} = \varnothing$;

$\boldsymbol{T}$ – конечное множество входных символов, $\boldsymbol{T} \subseteq \boldsymbol{U}$;

$\boldsymbol{F}$ – множество финальных состояний управляющего устройства, $\boldsymbol{F} \subseteq \boldsymbol{Q}$;

q_s – стартовое состояние управляющего устройства, s – мнемонический знак начала (*start*) $q_s \in \boldsymbol{Q}$;

q_f – финальное состояние управляющего устройства, f – мнемонический знак конца (*finish*) $q_f \in \boldsymbol{Q}$;

$\vdash$ – маркер начало ленты, $\vdash \in \boldsymbol{U}$, $\vdash \notin \boldsymbol{T}$;

ϕ – символ заполнитель пустой ячейки ленты, $\phi \in \boldsymbol{U}$, $\phi \notin \boldsymbol{T}$;

$\Lsh$ – символ, показывающий передвижение головки налево;

$\Rsh$ – символ, показывающий передвижение головки направо;

$\|$ – символ, показывающий непередвигаемость головки;

δ – функция перехода может быть не определено для некоторых аргументов и отображает множество $(\boldsymbol{Q}\backslash\boldsymbol{F})\times\boldsymbol{U}$ в множество $\boldsymbol{Q}\times(\boldsymbol{U}\backslash\{\phi\})\times\{\Lsh, \Rsh, \|\}$, т.е.

$$\delta: (\boldsymbol{Q}\backslash\boldsymbol{F})\times\boldsymbol{U}\rightarrow\boldsymbol{Q}\times(\boldsymbol{U}\backslash\{\phi\})\times\{\Lsh, \Rsh, \|\}$$

В недетерминированной машине Тьюринга функция перехода отображает множество $(\boldsymbol{Q}\backslash\boldsymbol{F})\times\boldsymbol{U}$ в множество всех подмножеств множества $\boldsymbol{Q}\times(\boldsymbol{U}\backslash\{\phi\})\times\{\Lsh, \Rsh, \|\}$, т.е.

$$\delta: (\boldsymbol{Q}\backslash\boldsymbol{F})\times\boldsymbol{U}\rightarrow\vartheta(\boldsymbol{Q}\times(\boldsymbol{E}\backslash\{\phi\})\times\{\Lsh, \Rsh, \|\}),$$

где $\vartheta(\boldsymbol{Q}\times(\boldsymbol{E}\backslash\{\phi\})\times\{\Lsh, \Rsh, \|\})$ – множество всех подмножеств множества $\boldsymbol{Q}\times(\boldsymbol{U}\backslash\{\phi\})\times\{\Lsh, \Rsh, \|\}$.

При работе МТ совершает элементарные шаги: считывание и запись символов, сдвиг головки на ячейку влево или вправо, а также переход управляющего устройства в следующее состояние, которым может быть новое состояние или старое состояние. За один шаг своей работы МТ способна выполнить следующие действия:

- управляюще устройство принимает текущее состояние;
- головка считывает символ в текущей ячейке;
- в зависимости от текущего состояния управляющего устройства и символа в текущей ячейке головка записывает в эту ячейку символ (быть может, совпадающий с прежним символом);
- головка сдвигается на ячейку влево или вправо, либо не двигается с места;
- управляющее устройство переходит в новое состояние или остается в своем старом состоянии;

- если управляющее устройство попадает в финальное состояние, то МТ завершает свою работу и останавливается.

Пусть заданы МТ $\boldsymbol{M}$ и его $q \in \boldsymbol{Q}$ – текущее состояние, $q_s \in \boldsymbol{Q}$ – стартовое состояние, $q_f \in \boldsymbol{Q}$ – финальное состояние (здесь s и f мнемонические знаки начала и конца), $\tau \in (\boldsymbol{U} \setminus \{\phi\})^*$ – непустая часть ленты, а $i \in \boldsymbol{N}$ – натуральное число, показывающее расстояние головки от левого конца цепочки τ. Тогда следующую тройку (q, τ, i) называют *конфигурацией* этой машины Тьюринга.

Далее, пусть $t_1 t_2 \ldots t_n \in T^*$ – входная цепочка, $(q, t_1 t_2 \ldots t_n, i)$, $1 \le i \le n+1$ – конфигурация машины Тьюринга $\boldsymbol{M}$. Тогда если:

1) $\delta(q, t_i) = (q', z, \leftarrow)$, $z \in \boldsymbol{T}$ $2 \le i \le n$, то шаг машины $\boldsymbol{M}$ может быть записан как отношение $(q, t_1 t_2 \ldots t_n, i) \models_{\boldsymbol{M}} (q', t_1 t_2 \ldots t_{i-1} z t_{i+1} \ldots t_n, i-1)$, здесь считается, что головка машины $\boldsymbol{M}$ записывает в i-*ю* ячейку символ z и сдвигается на одну ячейку влево, но не левее левого конца ленты $\vdash$;

2) $\delta(q, t_i) = (q', z, \rightarrow)$, $z \in \boldsymbol{T}$ $1 \le i \le n$, то шаг машины $\boldsymbol{M}$ может быть записан как отношение $(q, t_1 t_2 \ldots t_n, i) \models_{\boldsymbol{M}} (q', t_1 t_2 \ldots t_{i-1} z t_{i+1} \ldots t_n, i+1)$, здесь считается, что головка машины $\boldsymbol{M}$ записывает в i-*ю* ячейку символ z и сдвигается на одну ячейку вправо от нее;

3) $\delta(q, t_i) = (q', z, \|)$, $z \in \boldsymbol{T}$ $1 \le i \le n$, то шаг машины $\boldsymbol{M}$ может быть записан как отношение $(q, t_1 t_2 \ldots t_n, i) \models_{\boldsymbol{M}} (q', t_1 t_2 \ldots t_{i-1} z t_{i+1} \ldots t_n, i)$, здесь считается, что головка машины $\boldsymbol{M}$ записывает в i-*ю* ячейку символ z и остается на прежнем месте.

Если $i = n+1$, то головка машины $\boldsymbol{M}$ считывает специальный символ ϕ. При этом если:

1) $\delta(q, \phi) = (q', z, \leftarrow)$, то шаг машины $\boldsymbol{M}$ запишется как следующее отношение $(q, t_1 t_2 \ldots t_n, n+1) \models_{\boldsymbol{M}} (q', t_1 t_2 \ldots t_{i-1} z t_{i+1} \ldots t_n, n)$;

2) $\delta(q, \phi) = (q', z, \rightarrow)$, то шаг машины $\boldsymbol{M}$ запишется как отношение $(q, t_1 t_2 \ldots t_n, n+1) \models_{\boldsymbol{M}} (q', t_1 t_2 \ldots t_{i-1} z t_{i+1} \ldots t_n, n+2)$;

3) $\delta(q, \phi) = (q', z, \|)$, то шаг машины $\boldsymbol{M}$ запишется как отношение $(q, t_1 t_2 \ldots t_n, n+1) \models_{\boldsymbol{M}} (q', t_1 t_2 \ldots t_{i-1} z t_{i+1} \ldots t_n, n+1)$.

В случае, если известна МТ о которой идет речь, то букву $\boldsymbol{M}$ при отношении $\models_A$ можно опустить.

Как обычно, можно определить k-тую степень отношения $\models$: если между заданными двумя конфигурациями машины C_1 и C_k ($k \geq 1$) найдутся последовательность конфигурации $C_1, C_2, \ldots, C_k$ и между ними установлены отношения $C_1 \models C_2 \models \ldots \models C_k$, то этот факт можно записать как $C_1 \models^k C_k$.

Далее, если для некоторых $\tau \in T^*$, $\alpha \in U^*$ и любого числа $i \geq 1$ или $i \geq 0$ выполняется отношение $(q_s, \tau, 1) \models^i (q, \alpha, i)$, то его можно записать как отношение $(q_s, \tau, 1) \models^+ (q, \alpha, i)$ или $(q_s, \tau, 1) \models^* (q, \alpha, i)$, которое обозначает транзитивное замыкание или рефлексивное и транзитовное замыкание отношения $\models$ соответственно.

Язык $\boldsymbol{L}(\boldsymbol{M})$, распознаваемый МТ определяется как

$$\boldsymbol{L}(\boldsymbol{M}) \rightleftharpoons \{\tau:\ \tau \in \boldsymbol{T}^*\ \&\ (q_s, \tau, 1) \models^* (q_f, \alpha, i)\ \&\ \alpha \in \boldsymbol{U}^* \&\ i \in \boldsymbol{N}\}$$

Считается, что МТ $\boldsymbol{M}$, допускающая язык $\boldsymbol{L}(\boldsymbol{M})$, останавливается на каждом допустимом входе (т. е., допустив цепочку, машина прекращает работу). С другой стороны, возможно, что МТ $\boldsymbol{M}$ не остановится, если входная цепочка не принадлежит языку $\boldsymbol{L}(\boldsymbol{M})$.

Примеры V.2.1:

1. Язык $\{t^n b^n:\ n \geq 1\}$ распознается МТ $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{U}, \boldsymbol{T}, q_0, \boldsymbol{F}, \vdash, \leftarrow, \rightarrow, \|, \delta>$, где $\boldsymbol{Q} = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, $\boldsymbol{T} = \{t, b\}$, $\boldsymbol{U} = \{t, b, \phi, x, y\}$, $\boldsymbol{F} = \{q_5\}$ и δ:
$\delta(q_0, t) = (q_1, x, \rightarrow)$, $\delta(q_1, t) = (q_1, t, \rightarrow)$, $\delta(q_2, y) = (q_2, y, \leftarrow)$, $\delta(q_3, y) = (q_3, y, \rightarrow)$, $\delta(q_4, t) = (q_4, t, \leftarrow)$, $\delta(q_1, y) = (q_1, y, \rightarrow)$, $\delta(q_2, x) = (q_3, x, \rightarrow)$, $\delta(q_3, \phi) = (q_5, y, \rightarrow)$, $\delta(q_4, x) = (q_0, x, \rightarrow)$, $\delta(q_1, b) = (q_2, y, \leftarrow)$, $\delta(q_2, t) = (q_4, t, \leftarrow)$.
Последовательность действий для входной цепочки $\tau = ttbb$ такова:
$(q_0, ttbb, 1) \models (q_1, xtbb, 2) \models (q_1, xtbb, 3) \models (q_2, xtyb, 2) \models (q_4, xtyb, 1) \models$
$(q_0, xtyb, 2) \models (q_1, xxyb, 3)\ (q_2, xxyy, 2) \models (q_3, xxyy, 3) \models (q_1, xxyb, 4) \models$
$(q_2, xxyy, 3) \models (q_3, xxyy, 4) \models (q_3, xxyy, 5) \models (q_5, xxyyy, 6)$.

МТ для вычисления функции $U(n) = n+1$ задается как $\boldsymbol{Q} = \{q_s, q_t, q_f\}$, $\boldsymbol{T} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, q_0 - стартовое состояние, q_f – финальное состояние, а функция перехода δ в нижней таблице V.2.1, где столбец a реализует увеличение цифры в текущей ячейке на 1. Команда $\delta(q_s, 9) = (q_s, 0, \leftarrow)$ учитывает возникновение единицы переноса в старший разряд и при этом состояние q_s сохраняется. Именно в этом

состоянии мы увеличиваем цифру, в очередной ячейке на единицу. Команда $\delta(q_s, \phi) = (q_f, 1, \leftarrow)$ учитывает тот случай, когда в результате переноса разрядность числа возрастает на единицу. Последовательность команд в столбце *b*) обеспечивает соблюдение правила расположения результата.

Таблица V.2.1. Последовательность команд для реализации функции $U(n)=n+1$.

№ команды	***a***	***b***
1	$\delta(q_s, 0) = (q_t, 1, \Vert)$	$\delta(q_t, 0) = (q_t, 0, \leftarrow)$
2	$\delta(q_s, 1) = (q_t, 2, \Vert)$	$\delta(q_t, 1) = (q_t, 1, \leftarrow)$
3	$\delta(q_s, 2) = (q_t, 3, \Vert)$	$\delta(q_t, 2) = (q_t, 2, \leftarrow)$
4	$\delta(q_s, 3) = (q_t, 4, \Vert)$	$\delta(q_t, 3) = (q_t, 3, \leftarrow)$
5	$\delta(q_s, 4) = (q_t, 5, \Vert)$	$\delta(q_t, 4) = (q_t, 4, \leftarrow)$
6	$\delta(q_s, 5) = (q_t, 6, \Vert)$	$\delta(q_t, 5) = (q_t, 5, \leftarrow)$
7	$\delta(q_s, 6) = (q_t, 7, \Vert)$	$\delta(q_t, 6) = (q_t, 6, \leftarrow)$
8	$\delta(q_s, 7) = (q_t, 8, \Vert)$	$\delta(q_t, 7) = (q_t, 7, \leftarrow)$
9	$\delta(q_s, 9) = (q_t, 9, \Vert)$	$\delta(q_t, 9) = (q_t, 8, \leftarrow)$
10	$\delta(q_s, 9) = (q_s, 0, \leftarrow)$	$\delta(q_t, 9) = (q_t, 9, \leftarrow)$
11	$\delta(q_s, \phi) = (q_f, 1, \leftarrow)$	$\delta(q_t, \phi) = (q_f, \phi, \Vert)$

Язык ***L***, распознанный недетерминированной МТ, распознается и некоторой детерминированной МТ.

Это можно достичь, добавив к ленточному входному языку два специальных символа $\vdash$ и $\dashv$ для обозначения начала ленты и конца ленты. При этом если лента машины Тьюринга ограничена слева, т.е. символ $\vdash$ уже имеется, то достаточно добавить только символ $\dashv$. На ленте входная цепочка размещается между этими символами и имеет вид $\vdash\tau\dashv$. Функия перехода ЛОА подобна функции перехода для машины Тьюринга за исключением того, что если прочитан символ $\vdash$ или $\dashv$, то головка передвигается на одну ячейку вправо или влево.

Разумеется, справедливо, что любой язык, распознаваемый недетерминированным линейно ограниченным автоматом, распознается и некоторой детерминированной МТ.

Задания V.2.1.1:

1. Построить детерминированную машину Тьюринга с входным алфавитом $\{a\}$, вычисляющую функцию f, заданную соотношением $f(a^n) = a^{n+1}$.
2. Построить детерминированную машину Тьюринга с входным алфавитом $\{a\}$, вычисляющую функцию f, заданную соотношением $f(a^n) = a^{2n}$.
3. Построить детерминированную машину Тьюринга с входным алфавитом $\{a\}$, разрешающую язык пустой $\emptyset$.
4. Построить детерминированную машину Тьюринга с входным алфавитом $\{a\}$, разрешающую язык $\{a\}^*$.
5. Построить детерминированную машину Тьюринга с входным алфавитом $\{a\}$, разрешающую язык $\{a^n: n \geq 4\}$.
6. Построить детерминированную машину Тьюринга с входным алфавитом $\{a\}$, допускающую язык $\{a, aaa, aaaaaaa\}$.
7. Построить детерминированную машину Тьюринга с входным алфавитом $\{a\}$, вычисляющую функцию f, заданную соотношением $f(a^n) = \varepsilon$.

Вопросы V.2.1.1. Распознается ли следующий язык МТ:

1. $\{\tau c \tau : \tau \in \{a,b,c\}^*\}$?
2. $\{\tau \in \{a,b,c,d\}^* : |\tau|_a \neq |\tau|_b, |\tau|_c \neq |\tau|_d\}$?
3. $\{\tau \in \{a,b,c\}^* : |\tau|_a < |\tau|_b < |\tau|_c\}$?
4. Как в МТ задаются исходные данные?
5. Как в МТ задаются возможные результаты?
6. Как в МТ задаются промежуточные результаты?
7. Как в МТ задается правило начала работы А?
8. Как в МТ задается правило окончания работы?
9. Как в МТ извлекается результат?

10. Можно ли МТ строить из других МТ?

V.3. Свойства неограниченных языков

V.3.1. Эквивалентность неограниченных грамматик и машин Тьюринга

В этом разделе рассматриваются свойства неограниченных языков, которые порождаются неограниченными грамматиками (НГ) и распознаются машинами Тьюринга (МТ). Показывается эквивалентность НГ и МТ. Обсуждаются алгоритмические проблемы неограниченных языков. Показываются примеры, предлагаются задания и задаются вопросы [1-4,8,9,11-13,15-21,24-32].

Ниже доказывается, что язык распознается МТ тогда и только тогда, когда он порождается НГ.

Чтобы доказать *достаточность*, мы построим недетерминированную МТ, которая недетерминированно выбирает вывод в НГ и проверяет, совпадает ли результат этого вывода с входной цепочкой. Если совпадает, то машина распознает входную цепочку.

Чтобы доказать *необходимость*, мы строим грамматику, которая сначала порождает терминальную цепочку, а затем моделирует МТ на этой цепочке. Если цепочка принимается МТ, то цепочка преобразуется к терминальным символам, которые она представляет.

Лемма V.3.1. Если язык L порождается НГ G, т.е. $L=L(G)$, то существует МТ M, распознающая этот язык, т.е. $L. = L(M)$.

Доказательство. Пусть НГ $G = <N, T, P, S>$ порождает язык L, т.е. $L(G)= L$. Построим недетерминированную МТ M, так чтобы она распознавала этот язык, т.е. $L(M) = L$.

Пусть $M = <Q, U, T, q_0, F, \vdash, \nleftarrow, \twoheadrightarrow, \|, \delta>$ – недетерминированная МТ, где $U= N \cup T \cup \{\cap, \phi\}$, где $\cap$ – специальный символ для разделения цепочек, ϕ – специальный символ для заполнения пустых ячеек и $\cap$, $\phi \notin N \cup T$.

Сначала входная лента МТ M содержит цепочку $\tau \in T^*$. Затем она, сдвигая все символы τ на одну ячейку вправо, вставляет символ # на освободившееся место перед τ, а после цепочки τ вставляется цепочка

#*S*#, так что содержимым ленты становится цепочка #τ#*S*#. С этого момента МТ ***M*** будет недетерминированно моделировать вывод в НГ ***G***, начиная с нетерминального символа *S*. Каждая сентенциальная форма в выводе грамматики будет появляться по очереди между двумя последними ограничителями #. Если некоторый выбор движений приводит к цепочке терминалов, то она сравнивается с цепочкой τ. Если эти две цепочки равны, то МТ ***M*** распознает цепочку τ.

Точнее говоря, пусть в какой-то момент МТ ***M*** имеет на своей ленте цепочку вида $\#\tau\#A_1A_2...A_k\#$. Машина передвигает свою головку по цепочке $A_1A_2...A_k$, недетерминированно выбирая позицию i и константу r, равной максимальной длине левой части любого правила из множества ***P***. Затем машина 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 правил; если длина правой части правила не равна r, то машина может быть вынуждена сдвигать подцепочку $A_{i+r}A_{i+r+1}...A_k\#$ либо влево, либо вправо, чтобы освободить или заполнить место. При сдвиге вправо символ ϕ используется для временного заполнения освободившегося пространства. Из этого простого моделирования выводов в НГ ***G*** видно, что машина записывает на своей ленте цепочку вида #τ#λ#, где $\lambda \in \boldsymbol{T}^*$, точно тогда, когда $S \Rightarrow^*_{\boldsymbol{G}} \lambda$. Кроме того, если λ= τ, то МТ ***M*** распознает ***L***. Заметим, что для реализации проверки этого равенства опять пригодится символ ϕ. Что и требовалось доказать.

Лемма V.3.2. Если язык ***L*** распознается МТ ***M***, т.е. ***L***= ***L***(***M***), то существует НГ ***G***, порождающая этот язык ***L***, т.е. ***L***=***L***(***G***).

Доказательство. Пусть МТ $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{U}, \boldsymbol{T}, q_0, \boldsymbol{F}, \vdash, \nleftarrow, \nrightarrow, \|, \delta>$ распознает язык ***L***, т.е. ***L***(***M***) = ***L***. Теперь построим НГ ***G***, которая сначала недетерминированно порождает две копии представления некоторой цепочки из множества $\boldsymbol{T}^*$, затем моделирует действие МТ ***M*** на одной из этих копий. Если МТ ***M*** распознает цепочку, то НГ ***G*** превращает вторую копию в терминальную цепочку. Если МТ ***M*** не

распознает цепочку, то вывод в НГ $\boldsymbol{G}$ никогда не приводит к терминальной цепочке.

Без потери общности рассуждений предполагаем, что для каждого $q \in \boldsymbol{F}$ и $t \in \boldsymbol{T}$ значение $\delta(q, t)$ не определено.

Более детально, пусть задана НГ $\boldsymbol{G} = <\boldsymbol{N}, \boldsymbol{T}, \boldsymbol{P}, A_1>$, где $\boldsymbol{N} = \{[t, u] : t \in \boldsymbol{T} \cup \{\varepsilon\} \ \& \ u \in \boldsymbol{U}\} \cup \boldsymbol{Q} \cup \{A_1, A_2, A_3\}$, а $\boldsymbol{P}$ содержит правила:

(1) $A_1 \rightarrow q_0 A_2$;

(2) $A_2 \rightarrow [t, t] A_2$ для каждого $t \in \boldsymbol{T}$;

(3) $A_2 \rightarrow A_3$;

(4) $A_3 \rightarrow [\varepsilon, B] A_3$;

(5) $A_3 \rightarrow \varepsilon$;

(6) $q[t, c] \rightarrow [t, d]\ p$ для каждых $q \in \boldsymbol{Q}$, $t \in \boldsymbol{T} \cup \{\varepsilon\}$, $c \in \boldsymbol{U}$, $d \in \boldsymbol{U}$, $p \in \boldsymbol{Q}$ таких, что $\delta(q, c) = (p, d, \rightarrow)$;

(7) $[u, \varepsilon]\ q[t, c] \rightarrow p[u, \varepsilon]\ [t, d]$ для всех $u \in \boldsymbol{T} \cup \{\varepsilon\}$, $e \in \boldsymbol{U}$, $q \in \boldsymbol{Q}$, $t \in \boldsymbol{T} \cup \{\varepsilon\}$, $c \in \boldsymbol{U}$, $p \in \boldsymbol{Q}$, $d \in \boldsymbol{U}$ таких, что $\delta(q, c) = (p, d, \boldsymbol{L})$;

(8) $[t, c]q \rightarrow qtq$, $q[t, c] \rightarrow qtq$, $q \rightarrow \varepsilon$ для каждого $t \in \boldsymbol{T} \cup \{\varepsilon\}$, $c \in \boldsymbol{U}$, $q \in \boldsymbol{F}$.

Далее, используя вышеприведенные правила 1 и 2 для некоторого $t_i \in \boldsymbol{T}$, $1 \le i \le k$, получим вывод вида

$$A_1 \Rightarrow^* q_0[t_1, t_1]\ [t_2, t_2] \ldots [t_k, t_k] A_2$$

Предположим, что МТ $\boldsymbol{M}$ распознает цепочку $t_1 t_2 \ldots t_k$, тогда для некоторого m она использует не более, чем m ячеек справа от своего входа. Используя правило 3, затем m раз правило 4 и, наконец, правило 5, получим

$$A_1 \Rightarrow^* q_0[t_1, t_1]\ [t_2, t_2] \ldots [t_k, t_k]\ [\varepsilon, B]^m$$

Заметим, что начиная с этого момента и впредь только правила 6 и 7 могут использоваться пока не сгенерируется распознающее состояние. При этом первые компоненты ленточных символов в обозначениях нетерминалов $[(\boldsymbol{T} \cup \{\varepsilon\}) \times \boldsymbol{U}]$ никогда не изменяются, а вторые моделируют записи, производимые на ленте МТ $\boldsymbol{M}$.

Индукцией по числу шагов k МТ $\boldsymbol{M}$ можно показать, что если $(q_0, t_1 t_2 \ldots t_k, 1) \vdash_{\boldsymbol{M}}^* (q, X_1 X_2 \ldots X_s, r)$, то

$$q_0[t_1, t_1]\ [t_2, t_2] \ldots [t_k, t_k]\ [\varepsilon, B]^m \Rightarrow_{\boldsymbol{G}}^*$$

$$[t_1,X_1][t_2,X_2]...[t_{r-1},X_{r-1}]\ q[t_r,X_r]...[t_{k+m},X_{k+m}],$$
$$\text{где } t_1,t_2,...,t_k \in \boldsymbol{T};\ t_{k+1}=t_{k+2}=...=t_{k+m}=\varepsilon;$$
$$X_1, X_2, ..., X_{k+m} \in \boldsymbol{U};\ X_{s+1}=X_{s+2}=...=X_{k+m}=\phi.$$

База индукции. Пусть число шагов $k = 0$, тогда утверждение верно.

Индукционная гипотеза. Предположим, что утверждение выполняется для всех *k-1* шагов.

Индукционный переход. Пусть МТ $\boldsymbol{M}$ выполняет следующие k шагов:

$$(q_0, t_1t_2...t_k, 1) \models_{\boldsymbol{M}}^*$$
$$(q_1, X_1, X_2, ..., X_r, j_1) \models_{\boldsymbol{M}}^*$$
$$(q_2, Y_1Y_2...Y_s, j_2).$$

Тогда по индукционной гипотезе, применённой к первым k шагам, существует вывод вида

$$q_0[t_1, t_1][t_2, t_2]...[t_k, t_k][\ \varepsilon, B]^m \Rightarrow_{\boldsymbol{G}}^*$$
$$[t_1, X_1][t_2, X_2]... [t_{r-1}, X_{r-1}]q_1[t_{j1}, X_{j1}]... [t_{k+m}, X_{k+m}]$$

Судя по последнему шагу, должно быть $\delta(q_1, X_{j1}) = (\ q_2, Y_{j1}, d)$ и при этом $d =\!\!\leftarrow$, если $j_2 = j_1-1$ или $d = \rightarrow\!\!|$, если $j_2 = j_1+ 1$. Соответственно, при $d = \rightarrow\!\!|$ существует правило грамматики вида 6:

$$q_1[t_{j1}, X_{j1}] \rightarrow [t_{j1}, Y_{j1}]q_2,$$

а при $d =\!\!\leftarrow$ существует правило грамматики вида 7:

$$[t_{j1-1}, X_{j1-1}]q_1[t_{j1}, X_{j1}] \rightarrow q_2\ [t_{j1-1}, X_{j1-1}][t_{j1}, Y_{j1}]$$

Теперь $X_i = Y_i$ для всех $i \neq j_1$. Итак, еще один шаг вывода дает

$$q_0[t_1, t_1][t_2, t_2]...[t_k, t_k][\ \varepsilon, B]^m \Rightarrow_{\boldsymbol{G}}^*$$
$$[t_1, Y_1]q_2[t_{j2}, Y_{j2}]...[t_{k+m}, Y_{k+m}],$$

что доказывает предположение индукции. Далее, если $q \in \boldsymbol{F}$, то по правилам грамматики вида 8 можно получить вывод

$$[t_1, X_1] ... q[t_j, X_j] ... [t_{k+m}, X_{k+m}] \Rightarrow_{\boldsymbol{G}}^*$$
$$q\ t_1\ q\ t_2\ q... q\ t_k\ q \Rightarrow$$
$$_{\boldsymbol{G}}^*\ \ t_1t_2 ... t_k$$

Итак, НГ $\boldsymbol{G}$ может порождать цепочку $t_1t_2 ... t_k$, если $t_1t_2 ... t_k$ распознается МТ $\boldsymbol{M}$, т.е. доказано, что если $t_1t_2...t_k$ распознается МТ $\boldsymbol{M}$, то $t_1t_2...t_k$ принадлежит языку $\boldsymbol{L}(\boldsymbol{G})$, порождаемого НГ $\boldsymbol{G}$.

Для завершения доказательства необходимо показать, что все цепочки из $\boldsymbol{L}(\boldsymbol{G})$, распознается МТ. Индукцией доказывается, что если $A_1 \Rightarrow_{\boldsymbol{G}}^* \tau$, то цепочка τ распознается МТ $\boldsymbol{M}$.

Отметим, что любой вывод и, в частности, вывод $A_1 \Rightarrow_{\boldsymbol{G}}^* \tau$ может начинаться только по правилам 1– 5, дающим результат вида

$$A1 \Rightarrow_{\boldsymbol{G}}^* q_0[t_1, t_1][t_2, t_2]...[t_k, t_k][\ \varepsilon, B]^m.$$

Далее применяются правила 6 и 7, дающие в конце результат вида

$$[t_1, X_1][t_2, X_2]...\ q_1[t_{j1}, X_{j1}]...[t_{k+m}, X_{k+m}],$$

где $q_1 \in \boldsymbol{F}$, после чего правила 8 дадут $\tau = t_1 t_2 ... t_k$.

Покажем, что работа МТ $\boldsymbol{M}$ моделируются на участке вывода $q_0[t_1,$

$$t_1][t_2, t_2]...[t_k, t_k]\ [\varepsilon, B]^m \Rightarrow$$

$$_{\boldsymbol{G}}^* [t_1, Y_1][t_2, Y_2]...q[t_j, Y_j]...[t_{k+m}, Y_{k+m}],$$

где $q \in \boldsymbol{F}$. При этом, если такой вывод имеет место, то существуют шаги МТ $\boldsymbol{M}$ вида $(q_0, t_1, t_2, ..., t_k, 1) \models_{\boldsymbol{M}}^* (q, Y_1 Y_2 \ ... Y_{k+m}, j)$.

Докажем это утверждение индукцией по длине вывода l.

База индукции. Пусть $\boldsymbol{L} = 0$. Утверждение выполняется очевидным образом.

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

Индукционный переход. Пусть имеется вывод длиной $\boldsymbol{L} = n+1$. В общем случае имеем

$$q_0[t_1, t_1][t_2, t_2]...[t_k, t_k]\ [\varepsilon, B]^m \Rightarrow_{\boldsymbol{G}}^*$$

$$[t_1, X_1][t_2, X_2]...\ q_1[t_{j1}, X_{j1}]...[t_{k+m}, X_{k+m}] \Rightarrow_{\boldsymbol{G}}^*$$

$$[t_1, Y_1][t_2, Y_2]...q[t_j, Y_j]...[t_{k+m}, Y_{k+m}],$$

где $q \in \boldsymbol{F}$. Согласно индукционной гипотезе существует переход

$$(q_0, t_1, t_2, ..., t_k, 1) \models_{\boldsymbol{M}}^* (q_1, X_1 X_2 ... X_{k+m}, j_1).$$

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 $\delta(q_1, X_{j1}) = (q, Y_{j1}, d)$, где $d = \rightarrowtail$, если $j = j_1 + 1$, или $d = \leftarrowtail$, если $j = j_1 - 1$.

При этом $X_i = Y_i$ для всех $i \neq j_1$. Благодаря этим значениям функции δ МТ $\boldsymbol{M}$ совершает еще один шаг, переводящий в распознающую конфигурацию:

$$(q_0, t_1t_2...t_k, 1) \models_{\boldsymbol{M}}^* (q_1, X_1X_2...X_{k+m}, j_1) \models_{\boldsymbol{M}}^* (q, Y_1Y_2 ...Y_{k+m}, j),$$

где $q \in \boldsymbol{F}$.

Другими словами, показано, что цепочка $\tau = t_1\, t_2 \ldots t_k$ распознается МТ $\boldsymbol{M}$.

Итак, если $\tau \in \boldsymbol{L}(\boldsymbol{G})$, то цепочка τ распознается МТ $\boldsymbol{M}$.

Задания V.3.1. Построить МТ $\boldsymbol{M} = <\boldsymbol{Q}, \boldsymbol{U}, \boldsymbol{T}, \boldsymbol{q_0}, \boldsymbol{F}, \vdash, \nleftarrow, \nrightarrow, \Vert, \boldsymbol{\delta}>$, эквивалентный НГ $\boldsymbol{G}$, порождающей язык:

1. $\boldsymbol{L}(\boldsymbol{G}) = \{a^m b^n a^m b^n : \ m \geq 1 \ \& \ n \geq 1\}$?
2. $\boldsymbol{L}(\boldsymbol{G}) = \{a^k b^m c^n : \ k \geq 1 \ \& \ m \geq 1 \ \& \ n \geq 1\}$?
3. $\boldsymbol{L}(\boldsymbol{G}) = \{a^{k+n} b^{k-n}: \ k \geq 1 \ \& \ n \geq 1\}$?
4. $\boldsymbol{L}(\boldsymbol{G}) = \{\varepsilon\}$;
5. $\boldsymbol{L}(\boldsymbol{G}) = \{ a^{2n}: \ n \geq 1\}$;
6. $\boldsymbol{L}(\boldsymbol{G}) = \{a, aaa, aaaaaaa\}$;
7. $\boldsymbol{L}(\boldsymbol{G}) = \{a\}^*$;

Вопросы V.3.1.

1. Как строится машина Тьюринга, которая будет эквивалентна грамматике $\boldsymbol{G}$ с параметрами $\boldsymbol{N} = \{A, B, C, S\}$, $\boldsymbol{T}=\{a,b,c\}$, $\boldsymbol{P} = \{S \to aB, B \to aB, B \to b, B \to bC, C \to c, C \to cC, S \to aC, S \to a\}$, порождающей язык:

$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{ab^n c: \ n \geq 0\}?$$

2. Как строится машина Тьюринга, которая будет эквивалентна грамматике $\boldsymbol{G}$ с параметрами $\boldsymbol{N}=\{A, B, C\}$, $\boldsymbol{T} = \{a, b, c\}$, $\boldsymbol{P} = \{S \to aSBC \mid abC, CB \to BC, bB \to bb, bC \to bc, cC \to cc\}$, порождающей язык:

$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{(abc)^n: \ n >= 1\}?$$

3. Как строится машина Тьюринга, которая будет эквивалентна грамматике $\boldsymbol{G}$ с параметрами $\boldsymbol{N} = \{A, S\}$, $\boldsymbol{T} = \{a, b\}$, $\boldsymbol{P} = \{S \rightarrow A, A \rightarrow b, A \rightarrow cA\}$, порождающей язык:

$$\boldsymbol{L}(\boldsymbol{G}) \rightleftharpoons \{c^n b^{n+1}: \ n>=0\}.$$

V.3.2. Алгоритмические проблемы неограниченных языков

В этом параграфе рассматриваются алгоритмические проблемы неограниченных грамматик. Приводятся алгоритмически неразрешимые проблемы, имеющихся в неограниченных грамматиках и машинах Тьюринга (МТ), даются задания и вопросы [1-4,8,9,11-13,15-21,24-32].

Аналогично другим классам автоматов для МТ будем обсуждать следующие алгоритмические проблемы:

1. *Проблема пустоты* – для заданной МТ M выяснить $L(M)$ будет ли пустым языком, т.е. $L(M) = \varnothing$?

2. *Проблема бесконечности* – для заданной МТ M выяснить $L(M)$ будет ли бесконечным языком, т.е. $L(M) = \infty$?

3. *Проблема принадлежности* – для любой цепочки ξ выяснить принадлежит ли она языку $L(M)$, распознанному заданной МТ M, т.е. $\xi \in L(M)$?

4. *Проблема эквивалентности* – для любых двух МТ M_i и M_j ($i \neq j$) выяснить будут ли они эквивалентными, т.е. $L(M_i) = L(M_j)$?

5. *Проблема замкнутности* – при применении множественной операции к двум языкам, распознанным двумя разными МТ M_i и M_j выяснить тип результата будет таким же или нет?

6. *Проблема о самоприменимости* – существует ли МТ, которая может успешно завершиться на данных, представляющих собой формальную запись этой же МТ?

7. *Проблема остановки* – для заданного МТ M и заданных входных данных выяснить завершится ли когда-либо работа МТ с этими данными.

Доказательство алгоритмической неразрешимости некоторых проблем можно получить из следующих теорем:

Теорема V.3.1. Не существует МТ M, решающей проблему самоприменимости МТ M.

Теорема V.3.2. Не существует МТ $\boldsymbol{M}_0$, решающей проблему остановки для произвольной МТ $\boldsymbol{M}$.

Отсюда следует, что проблема остановки МТ алгоритмически неразрешимой, т. е. проблема определения результативности алгоритмов неразрешима.

Задания V.3.2:

1. Построить детерминированную машину Тьюринга с входным алфавитом $\{a\}$, допускающую язык $\{a^{3n+1} : n \geq 0\}$.
2. Построить детерминированную машину Тьюринга с входным алфавитом $\{a\}$, допускающую язык $\{a^{2n+1} : n \geq 0\}$.
3. Построить детерминированную машину Тьюринга с входным алфавитом $\{a,b,c\}$, допускающую язык $\{a^{k-m}b^m c^{n-m} : k \geq 1 \ \& \ m \geq 1 \ \& \ n \geq 1\}$.
4. Построить детерминированную машину Тьюринга с входным алфавитом $\{a,b\}$, допускающую язык $\{a^k b^k a^n b^n : k \geq 1 \ \& \ n \geq 1\}$.
5. Построить детерминированную машину Тьюринга с входным алфавитом $\{a,b\}$, допускающую язык $\{a^k b^k a^n b^n : k \geq 1 \ \& \ n \geq 1\}$.
6. Построить детерминированную машину Тьюринга с входным алфавитом $\{a,b,c\}$, допускающую язык $\{a^k b^{m-k} c^{n-m} : k \geq 1 \ \& \ m \geq 1 \ \& \ n \geq 1\}$.
7. Построить детерминированную машину Тьюринга с входным алфавитом $\{a,b,c\}$, допускающую язык $\{(a^k b^m) c^n : k \geq 1 \ \& \ m \geq 1 \ \& \ n \geq 1\}$.

Вопросы V.3.2:

1. В чем заключается проблема пустоты для машины Тьюринга?
2. В чем заключается проблема принадлежности для машины Тьюринга?
3. В чем заключается проблема остановки для машины Тьюринга?
4. В чем заключается проблема бесконечности для машины Тьюринга?
5. В чем заключается проблема эквивалентности для машины Тьюринга?
6. В чем заключается проблема замкнутости для машины Тьюринга?
7. Разрешима ли проблема о самоприменимости машины Тбюринга?

ЛИТЕРАТУРНЫЕ ИСТОЧНИКИ И ИНТЕРНЕТ РЕСУРСЫ

1. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. Т. 1: Синтаксический анализ. М.: Мир, 1978. - 612 с.
2. Бильгаева Н.Ц. Теория алгоритмов, формальных языков, грамматик и автоматов: Учебное пособие. Улан-Удэ: ВСГТУ, 2000.
3. Богаченко Н.Ф., Файзуллин Р.Т. Автоматы, грамматики, алгоритмы. Учебное пособие. Омск: Наследие. Диалог. 2006. -106 с.
4. Братчиков И. Л. Синтаксис языков программирования. М.: Мир, 1975. - 232 с.
5. Брауэр В. Введение в теорию автоматов. -М.: Радио и связь, 1987.
6. Волкова И.А., Руденко Т.В. Формальные грамматики и языки. Элементы теории трансляции. Издание второе (переработанное и дополненное) М: ВМиК МГУ, 1998.-62 с.
7. Гинзбург С. Математическая теория контекстно-свободных языков. М.: Мир, 1970. - 326 с.
8. Гладкий А. В. Формальные грамматики и языки. М.: Наука, 1973. - 368 с.
9. Гладкий А. В., Мельчук И. А. Элементы математической лингвистики. М.: Наука, 1969. - 192 с.
10. Глушков В.М. Синтез цифровых автоматов. М.: Физматгиз, 1962.
11. Гросс М., Лантен А. Теория формальных грамматик. М.: Мир, 1971. - 294 с.
12. Касьянов В. Н. Лекции по теории формальных языков, автоматов и сложности вычислений. - Новосибирск: НГУ, 1995. - 112 с.
13. Крючкова Е.Н. Теория формальных языков и автоматов. - Барнаул; 1996.
14. Куратовский К., Мостовский А. Теория множеств / Перевод с английского М. И. Кратко под редакцией А. Д. Тайманова. — М.: Мир, 1970. — 416 с.
15. Мелихов А.Н., Кодачигов В.И. Теория алгоритмов и формальных языков. - Таганрог; 1983.

16. Пентус А. Е., Пентус М. Р. Теория формальных языков: Учебное пособие. - М.: Изд-во ЦПИ при механико-математическом ф-те МГУ, 2004. - 80 с

17. Пентус А.Е., Пентус М.Р. Математическая теория формальных языков М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2006. - 248 с.

18. Рейуорд-Смит В. Дж. Теория формальных языков. Вводный курс. М.: Радио и связь, 1988. - 128 с.

19. Робин Хантер. Основные концепции компиляторов = The Essence of Compilers. — М.: «Вильямс», 2002. - С. 256.

20. Саломаа А. Жемчужины теории формальных языков. М.: Мир, 1986. - 159 с.

21. Соколов В. А., Кушниренко О. Б., Бадин Н. М. Формальные языки и грамматики. Задачи и упражнения. Ярославль: Ярославский государственный университет, 1993. - 55 с.

22. Столл Р. Р. Множества. Логика. Аксиоматические теории. / Перевод с английского Ю. А. Гастева и И. Х. Шмаина под редакцией Ю. А. Шихановича. — М.: Просвещение, 1968. — 232 с.

23. Трахтенброт Б. А., Барздинь Я. М. Конечные автоматы (поведение и синтез). М.: Наука, 1970. - 400 с.

24. Хомский Н., Миллер Дж. Введение в формальный анализ естественных языков // Под ред. А.А.Ляпунова и О.Б.Лупанова. - М.: Мир, 1965.

25. Хопкрофт Дж. Э., Мотвани Р., Ульман Дж. Д. Введение в теорию автоматов, языков и вычислений, 2-е изд. М.: Вильямс, 2002. - 528 с.

26. Rabin, M. O.; Scott, D. (April 1959). "Finite Automata and Their Decision Problems". IBM Journal of Research and Development 3 (2): 114–125. doi: 10.1147/rd.32.0114. Retrieved 2007-03-15.

27. https: //ru.wikipedia.org/wiki/Формальная_грамматика
28. http: //www.intuit.ru/studies/courses/...
29. http: //window.edu.ru/library/pdf2txt/905/27905/11127.
30. http: //www.cs.odu.edu/~toida/nerzic/390teched/web_course.html/ Introduction to Theoretical Computer Science Web Course.
31. http: //courses.cs.vt.edu/~cs4114/lectures/Formal Languages and Automata Theory Cource Lecture notes.
32. http: //www.cis.ksu.edu/~stough/forlan/book-and-slides.html/An Introduction to Formal Language Theory.
33. https: //ru.wikipedia.org/wiki/Множество.
34. http: //en.wikipedia.org/wiki/Automata_theory.

ШАРИПБАЙ Алтынбек Амирович

ТЕОРИЯ ЯЗЫКОВ И АВТОМАТОВ

Учебник

(Издание второе переработанное и дополненное)

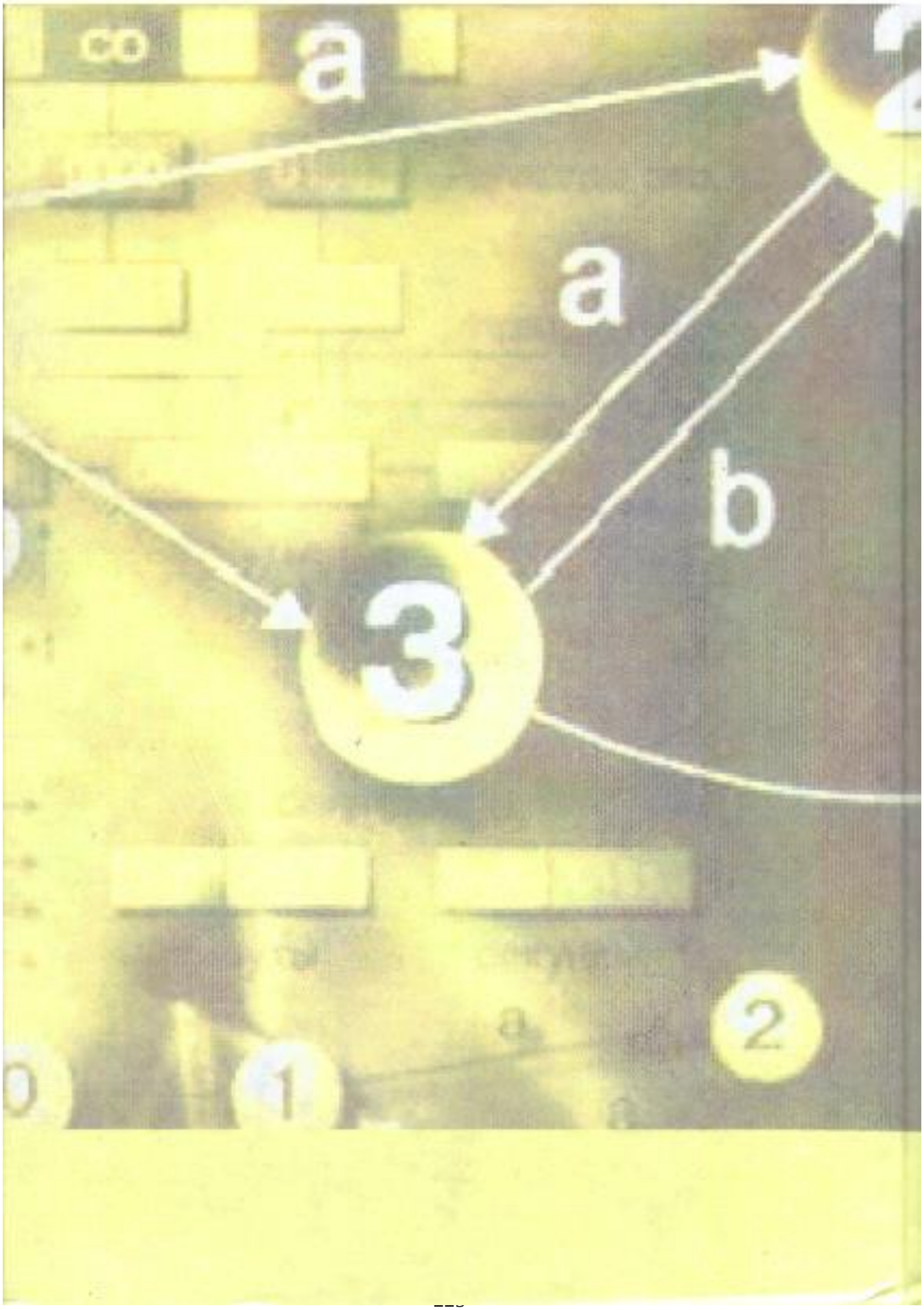
Бумага офсетная Формат 60x100 1/16
Плотность 80 гр/м^2 . Белизна 90%. Печать РИЗО.
Усл.печ.стр. 15.25. Объем 207.

Подготовлено к изданию и отпечатано
в издательстве “Эверо”
РК, Алматы, ул. Байтурсынова, 22
Тел.: +7 /727/ 233 83 89, 233 83 43,
233 80 45, 233 80 42
e-mail: evero08mail.ru

ШАРИПБАЙ Алтынбек Амирулы, доктор технических наук, профессор, академик Международной академии информатизации, академик Академии педагогических наук РК, лауреат государственной премии РК, директор НИИ «Искусственный интеллект» ЕНУ им.Л.Н.Гумилева (www.e-zerde.kz). Его научными интересами являются теоретические и прикладные проблемы информатики и информационных технологий: теория и технология программирования, автоматизированные системы, искусственный интеллект, компьютерная лингвистика и электронное обучение. В этой области им были разработаны методы формализации семантики продукционных и логических языков программирования, методы автоматизированной верификации программных и аппаратных средств. По результатам этих исследований он защитил докторскую диссертацию на тему «Верификация программных и аппаратных средств вычислительных машин и систем» по специальности «05.13.13 (05.13.11) - Вычислительные машины, системы и сети (Математическое, программное и техническое обеспечение)». Отдельные его научные результаты были внедрены в крупные научные центры: 1976-1979 годы «Транслятор с языка имитации космических систем» в Летно-испытательском институте, г. Жуковск; 1988-1989 годы «Система верификации цифровых схем» в Научно-производственном центре, г.Зеленоград; 1990-1991 годы «Система параллельного программирования для многопроцессорного вычислительного комплекса» в Научно-исследовательском центре электронной вычислительной техники, г. Москва.

Он опубликовал более 400 научных трудов, издал 5 учебников, 12 учебных пособий, 4 монографии и 5 терминологических и толковых словарей по информатике и вычислительной технике, получил более 30 свидетельств о государственной регистрации интеллектуальной собственности, участвовал в разработке многих государственных стандартов: 10 - в области информационных технологий, 9 - в области образования.

Под научным руководством А.Шарипбай подготовлены 4 докторов и 8 кандидатов наук, 8 доктора PhD по группе специальностей «Информатика, вычислительная техника и управления». Его научной школой создана математическая теория казахского языка, разработаны методы автоматизированного анализа и синтеза устных и письменных слов и предложений казахского языка, предложена технология создания электронных учебных изданий и др. Эти научные результаты в 1994-2014 годы были применены для создания многих электронных учебных изданий, системы дистанционного обучения казахскому языку, системы распознавания и синтеза казахской речи и других автоматизированных систем, в том числе учетных и экспертных систем, внедренных в различных государственных и негосударственных структурах и организациях РК.


a
a
b
3
2
1