ОГЛАВЛЕНИЕ

COLLECTION PROGRAMMATION
Publiée sous la direction de L. Nolin

PUBLICATIONS DE L'INSTITUT DE PROGRAMMATION
DE LA FACULTÉ DES SCIENCES DE PARIS

NOTIONS SUR LES GRAMMAIRES FORMELLES

PAR

Maurice GROSS et André LENTIN
Institut Blaise-Pascal

PARIS
GAUTHIER-VILLARS
1967

М. Гросс, А. Лантен

ТЕОРИЯ ФОРМАЛЬНЫХ ГРАММАТИК

Перевод с французского
И. А. МЕЛЬЧУКА

Под редакцией
А. В. ГЛАДКОГО

ИЗДАТЕЛЬСТВО «МИР»
Москва 1971

УДК 450 512 8

Книга М Гросса и А Лантена посвящена одной из наиболее важных областей математической лингвистики — теории формальных грамматик Хомского. В первой части вводятся необходимые понятия из алгебры, математической логики и теории алгоритмов Во второй рассматриваются некоторые классы формальных языков; третья часть посвящена алгебраической трактовке языков и их свойств.

Написанная на достаточно высоком уровне строгости, книга в то же время является сравнительно легкой для чтения Она будет полезна широкому кругу читателей: математикам, желающим ознакомиться с математической лингвистикой, специалистам по программированию и вычислительной математике, лингвистам и всем специалистам, работающим в смежных областях.

Редакция литературы по математическим наукам

Инд. 2-2-3, 7-1-1
8-71

ПРЕДИСЛОВИЕ РЕДАКТОРА ПЕРЕВОДА

Предлагаемая вниманию читателя книга М. Гросса и А. Лантена — одно из первых в мировой литературе руководств по теории формальных грамматик. По замыслу авторов эта книга предназначается для первоначального изучения указанной теории и должна быть доступна читателю с минимальной математической подготовкой; поэтому ее содержание ограничено основными понятиями и теоремами, а изложение носит элементарный характер. В то же время — и это очень существенное достоинство книги — теория формальных грамматик излагается в ней не изолированно, а в широком контексте понятий теории алгоритмов и теории автоматов (а также некоторых алгебраических концепций), что позволяет читателю более глубоко понять содержание самой теории грамматик и составить представление о ее месте в кругу родственных математических дисциплин. Необходимые понятия (машины Тьюринга, комбинаторные системы, конечные автоматы и т. д.) излагаются в самой книге, так что она может служить и для ознакомления с первоначальными (но только первоначальными) понятиями теории алгоритмов и теории автоматов.

Вообще отбор и расположение материала представляются в целом весьма удачными. Здание книги спроектировано авторами превосходно. Если бы вдобавок ими была проявлена высокая требовательность к выбору материала, книга была бы безупречной. Но, к сожалению, это не так: в ряде мест качество отделочного, а кое-где и основного строительного материала оставляет желать лучшего. Доказательства в книге, как правило, неформальные; при указанном характере книги это, конечно, правильно, но иногда стремление к неформальности и простоте заводит авторов слишком далеко, и тогда теряется ясность, а то и корректность изложения. Попадаются и просто небрежно проведенные и даже ошибочные рассуждения, а также не вполне согласованные между собой места. Редактор и переводчик старались по мере сил испра-

вить эти дефекты — иногда с помощью примечаний, иногда посредством незначительных сокращений, а чаще путем внесения исправлений в текст (без специальных оговорок). Трудно судить, в какой мере нам это удалось, но, во всяком случае, мы устранили все замеченные нами неточности и несогласованности. Примечания сделаны и в ряде мест, так или иначе требовавших пояснения.

Можно надеяться, что книга окажется полезной специалистам различных профилей, которые пожелают ознакомиться с основами теории формальных грамматик, — математикам, специалистам по программированию, лингвистам (по крайней мере тем из них, которые не боятся читать элементарно написанные математические книги), а также специалистам в области автоматической обработки информации.

А. Гладкий

ОТ АВТОРОВ

Настоящая книга написана на основе лекций, читанных авторами в ряде мест, из которых мы назовем здесь следующие: Центр количественной лингвистики факультета точных наук Парижского университета (созданный по инициативе покойного профессора Фавара); кафедра вычислительной математики того же факультета (профессор Рене де Поссель); кафедра математической физики Тулузского университета (профессор М. Лоде): курс для аспирантов по специальности «Обработка информации»; Отделение лингвистики Пенсильванского университета (профессор З. С. Хэррис); Институт программирования факультета точных наук Парижского университета.

Задуманная и написанная как учебник, эта книга не претендует на научную оригинальность. Значительную часть материала мы почерпнули из таких фундаментальных, ныне уже классических работ, как книга М. Дэвиса [1958] и ряд статей Н. Хомского (в первую очередь см. Хомский [1963]). Перечислять все источники многочисленных заимствований, вполне естественных в учебнике, вряд ли целесообразно, и мы ограничимся указанием двух, наиболее полезных для нас. Это лекции Ж. Питра, читавшиеся им в упомянутом выше Центре количественной лингвистики, и работы М. Нива, посвященные теории кодирования и преобразованиям формальных языков.

В настоящее время теория формальных грамматик привлекает внимание людей с самыми разными интересами и с самой разной подготовкой. Неоднородность круга наших потенциальных читателей поставила нас перед выбором — на какую подготовку ориентироваться при изложении материала. Мы решили сделать изло-

жение максимально доступным, и это иногда заставляло нас жертвовать подлинной математической строгостью ради большей наглядности и интуитивной простоты. По тем же соображениям мы старались не приводить доказательств, требующих сложной техники; некоторые из них вынесены в упражнения, относительно других мы ограничились ссылками на соответствующие работы.

М. Гросс
А. Лантен

ПРЕДИСЛОВИЕ

Традиционная лингвистика считала очень важным различие между «частными грамматиками», в компетенцию которых входило изучение специфических особенностей конкретных языков, и «универсальной грамматикой», занимавшейся общими свойствами человеческого языка как такового. С относительно недавнего времени указанное различие вновь стало привлекать внимание лингвистов. При этом оживление интереса к универсальной грамматике совпало с развертыванием серьезных исследований в той области математической лингвистики, которую иногда называют «алгебраической лингвистикой» — в отличие от статистических исследований языка или вероятностного моделирования речевого поведения. Такое совпадение представляется вполне естественным по целому ряду соображений.

Во-первых, алгебраическая лингвистика занимается изучением формальных свойств естественного языка вообще, отвлекаясь от того, как именно эти свойства реализуются в тех или иных конкретных языках. При таком понимании алгебраической лингвистики она в принципе не отличима от универсальной грамматики. Правда, на практике известное различие между ними существует: для универсальной грамматики характерна более эмпирическая направленность исследований, для алгебраической лингвистики — строго математический анализ формальных объектов, возникающих на базе изучения конкретных языков. Это различие отчасти свидетельствует о незрелости данной области, отчасти же отражает расхождение личных интересов и склонностей тех или иных ученых. Если данное различие удастся существенно ослабить, то появится надежда на создание настоящей математической теории языка, с помощью которой можно будет чисто абстрактным образом изучать класс систем, определяемый на основе принципов универсальной грамматики, — класс «возможных человеческих языков».

Во-вторых, тот факт, что одновременно со становлением алгебраической лингвистики возродился интерес к универсальной грамматике, объясняется также и всем ходом развития лингвистики за последнее столетие. К середине XIX века изучение универсальной грамматики (grammaire générale et raisonnée) пришло в полный упадок. Ее ценность и научная правомерность были поставлены под сомнение. Лингвисты в это время обратились к сравнительно-

историческим исследованиям, а позже к анализу языка в рамках оказавшегося весьма плодотворным «структуралистского», или «дескриптивистского», подхода. Размышления относительно grammaire générale et raisonnée стали казаться слишком умозрительными, оторванными от реальных языковых фактов. Было обнаружено, что в универсально-грамматических построениях сильно недооценивалось разнообразие языковых структур, а создававшееся при этом представление о строении языка вообще искажало картину наблюдаемых явлений конкретных языков. Следствием этого явилось разделявшееся рядом лингвистов мнение, будто естественные языки могут отличаться друг от друга чем угодно и насколько угодно, что никаких существенных общих ограничений на форму возможных человеческих языков нет вообще. В то же самое время в фокусе внимания лингвистов оказалась звуковая сторона речи, изучение которой до тех пор считалось чем-то вспомогательным, подчиненным задаче исследования более глубинных, в особенности синтаксических, свойств языка. Классическая лингвистика интересовалась в первую очередь тем свойством языка, которое можно было бы назвать его «творческим аспектом». Здесь имеется в виду тот факт, что естественный язык располагает рекурсивными механизмами, благодаря которым говорящий может — независимо от внешних стимулов и внутренних состояний — выражать бесконечно много разных мыслей, чувств, намерений и т. д. Что же касается современной науки о языке, то многие лингвисты бихейвиористской ориентации вообще отрицают «творческий аспект» в языке, рассматривая язык как систему навыков, сеть связей типа «стимул — реакция» и т. п. Другие, как например Ф. де Соссюр, признают «творческий аспект», но относят его к речи (parole) и считают чем-то второстепенным или даже вообще выходящим за пределы лингвистических исследований (в силу того, что «творческий аспект», как им представляется, не удается описать строгими лингвистическими правилами).

В-третьих, можно указать еще одну и, возможно, главную причину сдвига интересов и изменения общего взгляда на язык. Дело в том, что до известного времени при изучении рекурсивных механизмов в синтаксисе естественных языков было просто не на что опереться: соответствующие формализмы еще не были созданы, а существовавший тогда аппарат не позволял достаточно четко и содержательно описывать эти механизмы. Основные необходимые понятия возникли и стали уточняться всего лишь 30—40 лет тому назад в ходе исследований по основаниям математики. Более глубокое понимание природы рекурсивных механизмов и алгоритмов вообще, достигнутое за последние десятилетия, позволило вернуться к рассмотрению творческого аспекта речевой деятельности и предпринять попытку точно описать механизмы, которыми располагает каждый язык и которые необходимы для его свободного

и непринужденного использования. В настоящее время дисциплина, занимающаяся изучением подобных механизмов, известна под названием «теории порождающих грамматик». Порождающая грамматика того или иного естественного языка — это система правил, задающая потенциально бесконечное множество предложений данного языка и одновременно сопоставляющая каждому предложению описание его структуры, отражающее его существенные фонетические, синтаксические и семантические свойства[1]). Построение порождающих грамматик стало возможным благодаря развитию математики; поэтому нет ничего удивительного в том, что интерес к формальным свойствам грамматик и к алгебраической лингвистике в целом пробудился именно в связи с этим новым подходом к исследованию языка.

По всем отмеченным вопросам можно было бы сказать еще очень много. Однако, по моему мнению, и без того ясно, что скептическое отношение структурной и дескриптивной лингвистики к возможностям универсальной грамматики ничем не оправдано. Более того, я считаю, что классическая лингвистика действовала слишком робко, постулируя универсальные условия, которым подчиняются все человеческие языки, и ограничения, налагаемые на их форму. По размышлении я пришел к выводу, что не без основания можно принять следующую гипотезу относительно лингвистики ближайшего будущего: каждый естественный язык будет рассматриваться как конкретная реализация некоторой абстрактной, и притом весьма жесткой схемы, допускающей лишь очень ограниченные классы грамматических способов, синтаксических структур и т. п.; при этом окажется, что есть сколько угодно «воображаемых» языков, не укладывающихся в такую схему и, стало быть, не являющихся возможными человеческими языками (в некотором психологически очень важном смысле), но в то же время в принципе способных передавать любое содержание, выразимое на любом из возможных человеческих языков. Если это предположение окажется верным, то можно ожидать, что центральной задачей лингвистической теории станет математическое исследование универсальной грамматики — grammaire générale et raisonnée — в охарактеризованном выше смысле. Сейчас еще рано утверждать что-либо определенное о возможности осуществления сделанного предположения; тем не менее, то, что мы уже знаем,

---

[1]) Строго говоря, здесь должна была бы идти речь не о *порождающих* грамматиках, а о *формальных* грамматиках, представляющих собой более широкий класс объектов: порождающие грамматики — это частный случай формальных грамматик (хотя и наиболее важный); существуют формальные грамматики иных типов — например, распознающие. Определение порождающей грамматики, которое дает здесь Н. Хомский, приложимо в действительности к любой формальной грамматике. — *Прим. ред.*

и то, что мы узнаём в настоящее время, позволяет нам на это надеяться.

В заключение я хотел бы снова подчеркнуть, что между математическими и эмпирическими исследованиями в той области, которая в конце концов может превратиться в математическую теорию универсальной грамматики, все еще имеются существенные расхождения. Та схема грамматического описания, которая представляется эмпирически обоснованной, т. е. опирается на известные факты конкретных языков, задает класс систем, пока еще слишком сложных для серьезного и плодотворного математического изучения. Кроме того, следует иметь в виду, что все предложения относительно этой универсальной схемы, которые можно сформулировать уже сейчас, с одной стороны, в высшей степени гипотетичны, а с другой — достаточно расплывчаты в целом ряде важных отношений. В то же время можно сослаться на интересные и содержательные опыты исследования гораздо более узких (т. е. более специальных) схем грамматического описания — самым известным примером является, по-видимому, теория так называемых контекстно-свободных языков; однако эти системы безусловно неадекватны с эмпирической точки зрения. Таким образом, математическая теория универсальной грамматики представляется делом будущего, а самое большее, что можно сказать о современных исследованиях в данной области, — это то, что они ориентированы на создание подобной теории. Формальное изучение естественных языков является, на мой взгляд, одним из наиболее увлекательных научных направлений наших дней. Если оно будет развиваться успешно, то не исключено, что в ближайшие годы будут заложены совершенно новые основы науки о языке.

Н. Хомский

Часть I
ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ ИЗ ЛОГИКИ И АЛГЕБРЫ

Глава I
СЛОВА (ЦЕПОЧКИ). ПОЛУГРУППЫ. ЯЗЫКИ

§ 1.1. СВОБОДНАЯ ПОЛУГРУППА

1.1.0. Математика и метаматематика

Записывая или читая математический текст, мы используем некоторое количество символов и соглашений, более или менее точно определенных и более или менее точно сформулированных; именно это позволяет математикам правильно понимать друг друга.

Однако один и тот же символ может употребляться в математике в разных смыслах. Так, ясно, что скобки играют различную роль в следующих выражениях:

$$\text{функция } f(x),$$
$$y = (m+1)x^2 - 2mx + 2m - 3,$$
$$\text{пара } (x, y).$$

Ввиду неоднозначности символов может возникнуть, например, следующий вопрос: почему дистрибутивность умножения относительно сложения записывается в виде

$$(a+b)(c+d) = ac + ad + bc + bd,$$

а если мы имеем некоторую операцию $\circ$, дистрибутивную относительно некоторой другой операции $*$, то соответствующее выражение необходимо записывать более точно, т. е. сохраняя скобки:

$$(a \circ b) * (c \circ d) = (a \circ c) * (a \circ d) * (b \circ c) * (b \circ d)?$$

Подобные вопросы возникают в области обычной математики, но ответы на них следует искать в области *метаматематики.*

Вообще, человек, изучающий математику или пользующийся ею, нередко не проводит четкого различия между *математическим понятием* и *математическим методом* построения определения этого понятия и введения обозначений для его описания. Так, сплошь и рядом функцию смешивают с описывающей ее формулой. Впрочем, довольно часто сознательное неразличение формы и содержания, т. е. трактовка *семантики* как *синтаксиса*, позволяет

обнаруживать ошибки и восстанавливать смысл ошибочных или неоднозначных выражений.

Ясно, что машина не обладает тем запасом общих знаний и навыков, которые позволяют человеку выбирать правильную интерпретацию того или иного формального выражения. Поэтому, чтобы ввести в машину некоторое сообщение, мы должны пользоваться языком, в котором все определено совершенно строгим образом, вплоть до мельчайших деталей.

Для этого нам придется пойти на выучку к логикам, которые — к счастью для нас! — не дожидаясь появления электронных машин, начали *изучать метаматематику чисто математическими методами.*

Обращение к логике избавит нас от необходимости затрачивать значительные усилия на повторное открытие хорошо известных вещей.

В качестве первого шага мы приступим к рассмотрению «письма», начав с его самых простых аспектов — с употребления символов, которые можно располагать один за другим и образовывать из них слова, выражающие, быть может, тот или иной смысл.

Это и будет содержанием первой главы.

1.1.1. Базовые элементы

Пусть имеется непустое множество $\mathfrak{A}$, которое мы назовем *базой*, а его элементы — *базовыми элементами.* Чтобы как-то обозначить эти элементы, мы поставим в соответствие каждому из них графический символ, который и будет *«письменным» именем* этого элемента; а чтобы иметь возможность говорить о базовых элементах, мы сопоставим каждому из них *«устное» имя.* Само собой разумеется, что двум разным базовым элементам мы дадим разные имена.

Пример 1. Множество $\mathfrak{A}$ содержит один-единственный элемент, изображаемый символом | и обозначаемый словом «палочка».

Пример 2. Множество $\mathfrak{A}$ содержит два элемента, изображаемые символами ■ и □ и обозначаемые словами «черный» и «белый».

Символ, которым мы обозначаем базу (в нашем случае — символ $\mathfrak{A}$), не должен, естественно, использоваться для обозначения какого-либо элемента этого множества.

1.1.2. Конечные последовательности (цепочки)

Теперь мы рассмотрим некоторые образования, которые обычно называют размещениями элементов из $\mathfrak{A}$ с повторениями. Речь идет о *конечных последовательностях*, или *цепочках*, элементы которых суть вхождения элементов из $\mathfrak{A}$. Чтобы изобразить по-

добную цепочку графически, можно расположить на строчке (на бумажной или магнитной ленте и т. п.) вхождения символов, сопоставленных элементам $\mathfrak{A}$. Обычно в книгах последовательности символов всегда считаются ориентированными слева направо, если только в явной форме не оговорено противное.

Пример. Если база $\mathfrak{A}$ содержит три элемента, обозначенные соответственно через a, b и c, то запись

$$bacaba$$

изображает цепочку, образованную шестью элементами; первый элемент последовательности есть вхождение элемента b множества $\mathfrak{A}$ и т. д.

Удобно ввести в рассмотрение также *пустую* цепочку, не содержащую никаких вхождений.

Мы будем обозначать цепочки специальными символами, а сами цепочки заключать в одинарные кавычки (следя при этом, чтобы эти символы не использовались в качестве обозначений для каких-либо базовых элементов). Например, в предыдущем примере можно писать

$$A = \text{`}bacaba\text{'},$$

$$B = \text{`}acccab\text{'}.$$

Целесообразно выбрать один символ, например E, чтобы обозначать пустую цепочку:

$$E = \text{`\ '}.$$

Введенные здесь соглашения о способах записи цепочек позволяют получить предварительное представление об использовании *метаязыка* для описания некоторого языка; к этому вопросу мы вскоре еще вернемся.

Определим *длину* цепочки как целое число, равное числу вхождений элементов, образующих цепочку. Пустая цепочка будет иметь длину нуль. Длина цепочки обозначается с помощью вертикальных черточек:

$$|\,bacaba\,| = 6, \quad |\,A\,| = 6.$$

Аналогично определяется длина цепочки относительно одного или нескольких символов: '*bacaba*' имеет длину 3 относительно a (иначе — по a), длину 1 по c и т. п.

1.1.3. Конкатенация цепочек

Итак, пусть имеется множество $\mathfrak{A}$. Рассмотрим множество $\mathfrak{A}^*$ цепочек, образованных вхождениями элементов из $\mathfrak{A}$. Если даны цепочки A и B (в указанном порядке), то мы можем поставить в соответствие упорядоченной паре (A, B) цепочку C, полученную следующим образом: выпишем сначала подряд все вхождения из

A, а затем — непосредственно за ними — все вхождения из B. Мы будем говорить, что C получается *конкатенацией* A и B, и писать

$$C = AB.$$

Пример. Для $A = 'baca'$ и $B = 'ac'$ мы имеем $AB = 'bacaac'$ и $BA = 'acbaca'$.

Цепочка, полученная в результате конкатенации, имеет длину, равную сумме длин образующих ее цепочек.

Ясно, что конкатенация является всюду определенной и ассоциативной, но не коммутативной операцией; заметим, что для конкатенации существует единичный элемент, а именно пустая цепочка.

Множество, снабженное всюду определенной ассоциативной операцией, называется в алгебре *полугруппой*. Снабдив множество $\mathfrak{A}^*$ операцией конкатенации, мы превращаем его в *свободную полугруппу над* $\mathfrak{A}$, причем элементы $\mathfrak{A}$ являются *образующими* этой полугруппы.

Примеры. Если $\mathfrak{A} = \{a\}$, то свободная полугруппа изоморфна множеству степеней одной буквы a (с ассоциативным умножением).

Если $\mathfrak{A} = \{0, 1\}$, то цепочки множества $\mathfrak{A}^*$ можно трактовать как целые двоичные числа; тогда конкатенацию можно интерпретировать как некоторую довольно сложную арифметическую операцию.

1.1.4. Вопросы терминологии

В теории формальных грамматик терминология еще окончательно не установилась. Она колеблется в зависимости от авторов и от областей применения этой теории (искусственные языки, логика, естественные языки и т. д.).

Имея в виду исследование языков программирования, мы будем называть базу *абстрактным алфавитом*, состоящим из *абстрактных букв*, и говорить, что он представляется в данных условиях *конкретным алфавитом*, состоящим из *конкретных букв*. Цепочки в абстрактном алфавите мы будем называть *абстрактными словами*; абстрактные слова представляются *конкретными словами*[1]).

---

[1]) Таким образом, термин *слово* означает то же самое, что *цепочка*. В оригинале соответствующее понятие также обозначается в разных местах разными терминами (*mot*, *phrase*, иногда *séquence*) Такое словоупотребление связано со следующим обстоятельством. В алгебре и теории алгоритмов для обозначения конечных последовательностей символов (букв) уже довольно давно используется термин *слово* (нем *Wort*, англ. *word*, фр. *mot*) Однако в работах по математической лингвистике этого термина избегают, чтобы не создавать омонимии при лингвистической интерпретации, в которой слова естественного языка обычно соответствуют элементарным символам (буквам), а «словам» формальных

Так, в языке программирования Алгол-60 используется алфавит из 116 букв. Абстрактная буква определяется здесь множеством характеризующих ее свойств: например, абстрактная буква, которая играет роль начальной скобки перед набором команд, записывается с помощью конкретной буквы «**begin**» [1]); другая абстрактная буква, соответствующая логической импликации, записывается с помощью конкретной буквы «⊃» [2]). Не следует смешивать саму импликацию со значком ⊃ [3]). Подчеркнем, что для записи алгольных программ на перфоленту используется другой конкретный алфавит, а для записи соответствующей информации в машине — третий.

В естественных языках базу можно считать состоящей из *букв* или *фонем*, последовательности которых являются *словами*. Можно также рассматривать в качестве базы *словарь*, элементы которого суть *слова* (в обычном лингвистическом смысле); последовательности слов — это словосочетания, предложения и т. п. Здесь

языков отвечают словосочетания и предложения. Поэтому в работах по теории порождающих грамматик на английском языке вместо *word* обычно используется термин *string*, на русском языке — *цепочка*. В настоящем переводе также используется этот последний термин, но в разделах, относящихся к алгебре и теории алгоритмов, сохранен термин *слово*, так как в этом контексте термин *цепочка* звучал бы слишком непривычно — *Прим. ред.*

[1]) Буква «**begin**» принадлежит к международному алфавиту Алгола, описанному в официальном сообщении «Алгол-60» В СССР в этом же значении используется конкретная буква «**начало**».

О языке Алгол-60 см., в частности, «Алгоритмический язык Алгол-60 (пересмотренное сообщение)», изд-во «Мир», М., 1965. После выхода в свет этой книги появился новый вариант Алгола — Алгол-68, см официальное описание в журнале *Кибернетика* (Киев), № 6 (1969) и № 1 (1970). — *Прим перев.*

[2]) Авторы, по-видимому, считают, что во всех случаях, когда для обозначения импликации используется значок в форме подковы, обращенной закруглением вправо (а не, скажем, стрелка), или в качестве левой скобки перед набором команд — напечатанное полужирным шрифтом английское слово **begin** (а не русское слово **начало**, не какой-либо цифровой код, не курсивное *begin* и т. п.), можно говорить об одной и той же конкретной букве Такая трактовка представляется не вполне удачной. Понятие «значок ⊃» — это уже результат абстракции отождествления; при образовании этого понятия нам приходится отвлечься, в частности, от многочисленных графических особенностей конкретных значков данного вида Еще более это очевидно в отношении «буквы» **begin**. Естественнее всего, видимо, понимать конкретную букву как «физический предмет» (состоящий, например, из штрихов, нанесенных краской на бумагу). Некоторые конкретные буквы могут находиться в отношении «одинаковости», тогда они считаются представителями одной и той же абстрактной буквы. Разумеется, отношение «одинаковости» зависит от соглашения, мы вольны считать «а» и «*а*», «⊃» и «→», «**begin**» и «**начало**» представителями одной или разных абстрактных букв соответственно. Подробнее о конкретных и абстрактных буквах, алфавитах и словах см книгу А. А. Маркова «Теория алгорифмов», Л, 1954, стр. 7—15. — *Прим. ред*

[3]) Независимо от того, что понимается под значком «⊃» — абстрактная буква или представляющая ее конкретная. — *Прим ред.*

также необходимо различать графическое слово и то, что оно обозначает: известно, что слово «*собака*» не кусается.

Необходимо также различать букву «*а*», выступающую в роли базового элемента, и «*а*» как однобуквенное слово (союз «*а*»), как, например, в предложении «*Мари ест апельсин, а Жан стрижет ногти*».

Напротив, в теориях, где рассматриваются исключительно формальные свойства слов, безотносительно к какому бы то ни было смыслу, различать букву и однобуквенное слово, образованное этой буквой, незачем. Именно так обстоит дело в алгебраической теории свободных полугрупп.

§ 1.2. ОПЕРАЦИИ НАД СЛОВАМИ

1.2.1. Соотношения Туэ

Используемый в последующем изложении конкретный алфавит будет состоять из малых кириллических и латинских букв; слова будут обозначаться заглавными латинскими буквами, пустое слово — буквой E.

Слово '*картошка*' [1]) содержит два вхождения слова '*ка*': первое является началом слова '*картошка*', второе — его концом. Слово '*банан*' содержит два смежных вхождения слова '*ан*'. Соответствующие понятия нетрудно определить в общем виде.

Пусть P и Q — два разных слова (любое из них может быть пустым). Предположим, что некоторое слово A содержит вхождение слова P и потому представимо в виде

$$A = XPY.$$

Такому слову A можно поставить в соответствие слово B, полученное из A подстановкой вхождения Q вместо вхождения P.

Пример. Если P — слово '*ара*', а Q — слово '*ло*', то слову '*парад*' соответствует слово '*плод*'.

В этой главе мы будем рассматривать случай, когда, имея право замещать любое вхождение P вхождением Q, мы в то же время имеем право поступать и наоборот: замещать любое вхождение Q вхождением P.

Тогда мы будем писать

$$P \sim Q; \tag{1}$$

в предыдущем примере:

$$\text{'ара'} \sim \text{'ло'}.$$

Между словами могут иметь место одновременно несколько *соотношений* типа (1); например:

$$\text{'ара'} \sim \text{'ло'}; \quad \text{'ад'} \sim \text{'к'}.$$

[1]) В оригинале здесь и ниже в качестве примеров используются французские слова, которые мы заменили подходящими русскими — *Прим. перев.*

Подобные соотношения, называемые соотношениями Туэ (по имени впервые исследовавшего их норвежского математика), приводят к так называемым *ассоциативным исчислениям.*

Если слово B получается из слова A однократным применением одного соотношения, мы будем говорить, что A и B являются *смежными*; ясно, что в таком случае A тоже получается из B применением одного соотношения.

Так, в нашем примере

'*парк*' является смежным с '*парад*',

'*парад*' является смежным с '*плод*',

однако '*парк*' не является смежным с '*плод*'.

Чтобы получить транзитивное отношение, необходимо допустить несколько последовательных применений соотношений типа $P \sim Q$. Сформулируем строгое определение.

Определение. Пусть $\mathfrak{A}$ — некоторый алфавит и $R: P_1 \sim Q_1, \ldots, P_n \sim Q_n$ — система соотношений Туэ. Тогда мы будем говорить, что слово B_0 *соотносимо* со словом B_p, если существует последовательность слов $B_0, \ldots, B_p$, таких, что B_i является смежным с B_{i-1} для $i = 1, \ldots, p$.

Условимся, кроме того, считать, что любое слово соотносимо с самим собой. Тогда соотносимость будет отношением эквивалентности. Это позволяет нам, следуя существующей традиции, называть соотносимые слова *эквивалентными.* Так, в нашем примере слово '*парк*' эквивалентно слову '*плод*'. Отношение эквивалентности мы будем обозначать знаком «$\approx$» и писать: '*парк*' $\approx$ '*плод*' и т. п.

Теперь мы можем сформулировать следующую теорему (ее доказательство представляется очевидным):

Теорема. *Для любых слов X и Y из $A \approx B$ следует $XAY \approx XBY$.*

1.2.2. Проблема эквивалентности (слов)

Пусть имеется ассоциативное исчисление, заданное алфавитом $\mathfrak{A}$ и соотношениями

$$P_1 \sim Q_1, \ldots, P_n \sim Q_n.$$

Туэ сформулировал следующую фундаментальную проблему, известную под названием «*проблемы эквивалентности*»:

Для произвольной пары слов установить, являются они эквивалентными или нет.

1.2.3. Несколько примеров

Прежде всего мы дадим два определения, которые в дальнейшем не раз будут нам полезны.

Цепочка $\bar{A}$ называется *обращением* (*зеркальным образом*) цепочки A, если $\bar{A}$ состоит в точности из тех же вхождений, что и A, но взятых в обратном порядке, например:

$$A = \text{'ток'},\ \bar{A} = \text{'кот'} \quad \text{или} \quad A = \text{'roma'},\ \bar{A} = \text{'amor'}.$$

Цепочка A называется *симметричной*, если она совпадает со своим обращением; например: $B = $ '*шалаш*', $C = $ '*тортскофенефокстрот*'.

Рассмотрим следующее ассоциативное исчисление:

$$\mathfrak{A} = \{a, b\};\quad aa \sim E;\quad bb \sim E.$$

(Отметим попутно, что такие исчисления называются *нильпотентными*.)

Сократить слово A в этом исчислении — значит образовать новое слово B, смежное с A, причем B должно быть короче A.

Взяв некоторое слово за исходное, мы можем путем последовательных сокращений образовать ряд слов, который приведет нас к несократимому слову; оно может быть, в частности, пустым.

Пример. Возьмем в качестве исходного слова

$$A = a\,b\,a\,a\,a\,a\,b\,b\,a\,b\,a\,b\,b\,b\,a\,a\,b.$$

Один возможный для этого слова ряд последовательных сокращений отвечает следующей схеме:

$$\begin{array}{l} A \approx a\,b\,a\,a\,a\,a\,b\,b\,a\,b\,a\,b\,b\,b\,a\,a\,b, \\ \qquad\quad\ 2\ 2\quad 1\ 1\qquad\quad 6\ 6\quad 4\ 4 \\ \qquad\quad 3\qquad 3\qquad\qquad\quad 5\qquad 5 \\ A \approx a\,b\,a\,b\,a. \end{array}$$

Имеются и другие способы сократить это слово, например:

$$\begin{array}{l} A \approx a\,b\,a\,a\,a\,a\,b\,b\,a\,b\,a\,b\,b\,b\,a\,a\,b, \\ \qquad\quad\ 1\ 1\quad 3\ 3\qquad\quad 4\ 4\ 5\ 5 \\ \qquad\quad 2\qquad 2\qquad\qquad 6\qquad\qquad 6 \\ A \approx a\,b\,a\,b\,a. \end{array}$$

Интуитивно ясно, что результат не зависит от того, в каком порядке мы выполняем сокращения. Это видно из примера; доказательство же этого факта мы предоставляем читателю в качестве упражнения. Таким образом, в каждом классе эквивалентности существует ровно одно несократимое слово, которое естественно считать *каноническим представителем* этого класса.

Поэтому для данного ассоциативного исчисления проблема эквивалентности разрешима: любые два слова эквивалентны тогда и только тогда, когда они обладают одним и тем же каноническим представителем, т. е. когда они эквивалентны одному и тому же несократимому слову, которое нетрудно найти с помощью механизируемого процесса (процесса сокращения).

Однако далеко не всегда ситуация бывает столь благоприятной. Рассмотрим, например, тот же алфавит $\{a, b\}$ и соотношения

$$aaa \sim aa, \quad aba \sim aa, \quad bab \sim bb, \quad bbb \sim bb.$$

Из слова *ababa* можно получить или слово *aaa*, а затем *aa*, или *abba*. Слова *aa* и *abba* эквивалентны, несократимы и различны.

Проблема эквивалентности разрешима, когда в классах эквивалентности существуют инварианты.

«Универсальный» метод решения проблемы эквивалентности, который прежде всего приходит в голову, состоит в следующем: взяв произвольную пару слов S и T, последовательно образовать все слова, смежные с S, потом все слова, смежные с каждым из слов, полученных на первом шаге, и т. д., т. е., короче говоря, *перечислить* все слова, эквивалентные слову S, применяя сначала одно, потом два, потом три преобразования и т. д., пока мы не дойдем до T. Однако сколько бы преобразований ни было выполнено, тот факт, что T не найдено, еще ничего не означает: оно может быть получено после очередных преобразований. Таким образом, наш наивный «универсальный» метод не гарантирует нахождения решения.

Возникает вопрос: существует ли настоящий универсальный метод, позволяющий решать проблему эквивалентности слов? После того как мы уточним представление о методе решения вообще — точнее, введем понятие алгоритма, — можно будет показать, что ответ на этот вопрос является отрицательным.

1.2.4. Полугруппа классов

Пусть в некотором ассоциативном исчислении имеют место соотношения $S \approx S'$ и $T \approx T'$. Тогда (в силу теоремы, сформулированной в п. 1.2.1) мы имеем

$$ST \approx S'T \approx S'T',$$

откуда

$$ST \approx S'T'.$$

Таким образом, доказана следующая

Лемма. *Отношение эквивалентности в смысле Туэ сохраняется при операции конкатенации слов.*

Если отношение эквивалентности совместимо влево и вправо с полугрупповой операцией, его называют *конгруэнцией* [1]). Предыдущая лемма может быть сформулирована еще и так:

Эквивалентность в смысле Туэ есть конгруэнция на свободной полугруппе.

В алгебре широко используется следующая

[1]) Отношение эквивалентности $\sim$ называется совместимым влево (вправо) с операцией $\cdot$, если из $A \sim B$ следует $C \cdot A \sim C \cdot B$ ($A \cdot C \sim B \cdot C$). — *Прим. ред.*

Теорема. *Конгруэнция $\mathfrak{R}$, определенная на множестве M, снабженном ассоциативной бинарной операцией, определяет множество классов эквивалентности, или факторможество $M/\mathfrak{R}$, которое в свою очередь снабжено индуцированной бинарной ассоциативной операцией.* (Определение этой операции см. ниже в упр. 1.4.4.)

Если при этом исходная операция обладает единичным элементом, то класс этого элемента будет единичным элементом для индуцированной операции.

Доказательство этой фундаментальной теоремы намечено в упр. 1.4.4.

Для интересующего нас случая — эквивалентности Туэ на свободной полугруппе — только что сформулированное утверждение имеет следующий вид:

Теорема. *Классы, определяемые на свободной полугруппе эквивалентностью Туэ, образуют полугруппу относительно операции, индуцированной конкатенацией.*

Пример. $\mathfrak{A} = \{a, b\}$; $aa \sim E$, $bb \sim E$. Этот пример уже рассматривался в п. 1.2.3.

Здесь классы можно представлять в канонической форме с помощью сокращенных слов, в которых любые два последовательных вхождения обязательно являются вхождениями разных букв. Упорядочив эти сокращенные слова по возрастанию длин, — причем слова одинаковой длины упорядочиваются лексикографически, — мы получаем следующие классы:

$$E,\ a,\ b,\ ab,\ ba,\ aba,\ bab,\ abab,\ \ldots.$$

Пусть нам даны в некотором фиксированном порядке два класса; тогда для нахождения представителя композиции этих классов необходимо действовать в соответствии со следующими правилами:

1. Выполнить конкатенацию канонических представителей исходных классов в заданном порядке.
2. Рассмотреть вхождения «на стыке».
3. *Если* буквы «на стыке» различны, *операция закончена.*
   В противном случае: стереть два соседних вхождения одной и той же буквы.
4. *Если* результат равен E, *операция закончена. В противном случае*: перейти к правилу 2.

Ясно, что единичный класс будет получен тогда и только тогда, когда канонические представители являются обращениями друг друга.

Любой класс допускает единственное обращение, одно и то же влево и вправо, откуда — пары взаимно обратных слов:

$$E,\ E;\ a,\ a;\ b,\ b;\ ab,\ ba;\ aba,\ aba;\ bab,\ bab;\ abab,\ baba;\ \ldots.$$

Класс, имеющий своим представителем симметричное сокращенное слово, является своим собственным обращением.

Читатель уже заметил, по-видимому, что в данном случае полугруппа классов является группой.

Что касается ядра (множества слов класса E), то оно включает симметричные слова четной длины, а также слова, полученные конкатенацией этих последних, например:

$$baab,\quad abba,\quad bb.$$

§ 1.3. ЯЗЫКИ

1.3.1. Определение

Формальным языком, определенным на базовом множестве $\mathfrak{A}$, называется любое подмножество свободной полугруппы $\mathfrak{A}^*$, иначе — любое множество цепочек из элементов $\mathfrak{A}$. Для краткости мы будем говорить просто «язык», опуская прилагательное «формальный».

Примеры. Для любого $\mathfrak{A}$ множество слов четной длины является языком; множество слов нечетной длины также является языком, но другим: первый язык *замкнут относительно операции конкатенации*, а второй нет.

Для любого $\mathfrak{A}$ множество симметричных слов является языком, не замкнутым относительно конкатенации (кроме случая, когда $\mathfrak{A}$ состоит из одной буквы).

1.3.2. Один важный пример: языки Дика

Представим себе набор формул некоторого математического исчисления или программу, записанную на языке типа Алгола. Это будет текст, в котором, видимо, будут встречаться знаки, всегда употребляемые только попарно: левые и правые скобки всех видов (круглые, квадратные, фигурные, ломаные) или такие выражения, как **begin — end** и т. п. Выкинем из текста все, кроме знаков указанного типа. Получится новый текст, построенный по некоторым строгим правилам.

Пример. Возьмем следующую правильную алгебраическую формулу:

$$\{[(a+b)(c+d)-(a'+b')(c'+d')]^6-(abcd-a'b'c'd')^3\}^2-$$
$$-\{[(aa'+bb'+cc'+dd')^6+(abcda'b'c'd')^{3/2}]+A\}^2.$$

Указанным способом из нее можно получить выражение

$$\{[\,()\,()\,()\,()\,]\,()\,\}\{[\,()\,()\,]\}.$$

Подобные тексты дают представление об *ограниченных языках Дика*.

Интерес к языкам Дика объясняется (по крайней мере отчасти) тем, что они очевидным образом связаны со *скобочными структурами*, обычными для естественных и искусственных языков (логика, математика, языки программирования и т. д.).

Перейдем к точным определениям.

Определения. Пусть $\mathfrak{A} = \{a, a', b, b', \ldots\}$ — алфавит, состоящий из $2n$ букв, объединяемых в пары: a, a'; b, b'; ...; пусть E — пустое слово, и пусть даны соотношения Туэ:

$$aa' \sim E; \quad bb' \sim E; \ldots .$$

Слово X принадлежит *ограниченному языку Дика* над $\mathfrak{A}$ тогда и только тогда, когда оно эквивалентно в смысле этих соотношений пустому слову.

Аналогичным образом определяется *неограниченный язык Дика*; при этом берутся соотношения

$$aa' \sim a'a \sim E, \quad bb' \sim b'b \sim E, \ldots .$$

Примеры. Слово $abb'a'cc'$ принадлежит ограниченному языку Дика.

Слово $a'ab'cc'b$ принадлежит языку Дика.

1.3.3. Операции над языками

Пусть $\mathfrak{A}$ — некоторый алфавит и $\mathfrak{A}^*$ — свободная полугруппа над $\mathfrak{A}$. Поскольку (формальные) языки над $\mathfrak{A}$ являются *подмножествами* множества $\mathfrak{A}^*$, можно определить в *множестве подмножеств* из $\mathfrak{A}^*$ [это множество мы будем обозначать через $\mathfrak{P}(\mathfrak{A}^*)$] некоторые теоретико-множественные операции. Рассмотрим основные операции этого типа.

Объединение языков. Пусть даны два языка L_1 и L_2. *Объединением* этих языков (обозначение: $L_1 \cup L_2$) называется множество всех слов, принадлежащих хотя бы одному из языков.

Эта операция представляет собой обычное теоретико-множественное объединение; она коммутативна и ассоциативна:

$$L_1 \cup L_2 = L_2 \cup L_1,$$
$$(L_1 \cup L_2) \cup L_3 = L_1 \cup (L_2 \cup L_3).$$

Пересечение языков. При тех же условиях *пересечением* двух языков (обозначение: $L_1 \cap L_2$) называется множество всех слов, принадлежащих одновременно обоим языкам.

Эта операция представляет собой обычное теоретико-множественное пересечение; она тоже коммутативна и ассоциативна.

Дополнение. *Дополнением* языка L до $\mathfrak{A}^*$ (обозначение: $\mathfrak{A}^* \setminus L$) называется множество всех слов, принадлежащих $\mathfrak{A}^*$, но не принадлежащих L,

Само множество $\mathfrak{A}^*$ есть язык, дополнением которого до $\mathfrak{A}^*$ является пустой язык.

Подчеркнем, что язык, содержащий пустое слово E, не является пустым.

Пример.

$\mathfrak{A} = \{a, b\}$;

$L = \{a^m b^n \mid m \geqslant 1,\ n \geqslant 1\}$;

$\mathfrak{A}^* \setminus L = (L_1 \cup L_2 \cup L_3)$, где L_1 — множество всех слов, начинающихся с b, L_2 — множество всех слов, начинающихся с $a^m b^n a$, и $L_3 = \{a\}^*$, т. е. L_3 есть множество всех слов, состоящих только из a.

Умножение языков. При тех же условиях, что и выше, *произведением* двух языков (обозначение: $L_1 L_2$) называется множество всех слов, которые можно получить следующим способом: берется некоторое слово из L_1 и к нему присоединяется конкатенацией справа некоторое слово из L_2, т. е.

$$L_1 L_2 = \{X_1 X_2 \mid X_1 \in L_1,\ X_2 \in L_2\}.$$

Эта операция (называемая *умножением языков*) не совпадает с декартовым умножением; она ассоциативна, но не коммутативна.

Примеры. Пусть $\mathfrak{A} = \{a, b, c\}$. Рассмотрим язык $\{a\}$, состоящий из одного однобуквенного слова 'a'. Обозначим этот язык через a. Тогда произведение

$$a\mathfrak{A}^*$$

есть множество всех слов, начинающихся с a.

Вообще, $\mathfrak{A}\mathfrak{A}^*$ — это множество всех слов, начинающихся с некоторой буквы из $\mathfrak{A}$, т. е. множество всех *непустых слов*:

$$\mathfrak{A}\mathfrak{A}^* = \mathfrak{A}^* \setminus \{E\}.$$

Операция итерации (операция Клини). Поскольку операция умножения языков ассоциативна, мы можем возводить данный язык L в степень:

$$L^2 = LL;\quad L^3 = (LL)L = L(LL) \text{ и т. д.}$$

С. К. Клини предложил рассматривать объединение

$$E \cup L \cup L^2 \cup L^3 \cup \ldots \cup L^n \cup \ldots$$

всех последовательных степеней языка L [1]). Это объединение обозначается L^* и называется *итерацией* языка L.

Пример. Мы можем рассматривать алфавит

$$\mathfrak{A} = \{a,\ b,\ \ldots\}$$

как язык, состоящий из однобуквенных слов. Тогда $\mathfrak{A}^2$ — множество всех диграмм (двухбуквенных слов), $\mathfrak{A}^3$ — множество всех триграмм и т. д. Поэтому

$$E \cup \mathfrak{A} \cup \mathfrak{A}^2 \cup \mathfrak{A}^3 \cup \ldots$$

есть множество всех слов над $\mathfrak{A}$, т. е. $\mathfrak{A}^*$.

Этим и объясняется выбор обозначения $\mathfrak{A}^*$ для свободной полугруппы над $\mathfrak{A}$.

Операция обращения. Пусть дан язык $L \subset \mathfrak{A}^*$; через $\tilde{L}$ обозначается язык, состоящий из обращений всех слов языка L:

$$\tilde{L} = \{\tilde{X} \mid X \in L\}.$$

Эта операция является инволютивной (т. е. совпадает со своей обратной операцией):

$$\tilde{\tilde{L}} = L.$$

Кроме того, она связана с умножением языков следующим соотношением:

$$\widetilde{LM} = \tilde{M}\tilde{L}.$$

Пример. Язык $\mathfrak{A}^*\tilde{\mathfrak{A}}^*$ совпадает с $\mathfrak{A}^*$, поскольку $\tilde{\mathfrak{A}}^* = \mathfrak{A}^*$ и $E \in \mathfrak{A}^*$.

§ 1.4. УПРАЖНЕНИЯ

1. Какое соглашение мы неявно принимаем, записывая пустое слово как

$$E = \text{`\ '}?$$

Достаньте какое-либо описание Алгола-60 и изучите статус *пробела* в Алголе.

2. Можно ли сказать о некотором алфавите $\mathfrak{A}$, что он содержит «пустую букву»?

3. Пусть имеется алфавит

$$\mathfrak{A} = \{a,\ b,\ c,\ x,\ \equiv,\ \times\}.$$

[1]) По определению $L^0 = \{E\}$; вместо $\{E\}$ пишут обычно E. — *Прим. ред.*

Рассмотрим слово

$$`a \times b \times c \equiv x'.$$

Имеет ли смысл это выражение, если a, b, c и x интерпретируются как числа, $\times$ как символ умножения, а $\equiv$ как равенство? Сохраняет ли оно смысл при интерпретации a, b, c и x как свободных векторов в евклидовом пространстве, а $\times$ — как векторного произведения? Какой вывод, относящийся к конкатенации, можно сделать из сравнения этих двух случаев?

4. Рассмотрим множество $S = \{A, B, C, \ldots\}$, снабженное операцией *композиции* (которую мы обозначим точкой); на множестве S определена эквивалентность, совместимая слева и справа с этой операцией, т. е. *конгруэнция*.

1°) Обозначим через $\bar{A}$ класс элемента A, через $\bar{B}$ — класс элемента B и т. д. Доказать, что класс композиции $A \cdot B$ зависит только от $\bar{A}$ и $\bar{B}$; показать, что можно определить композицию классов следующим образом:

$$\bar{A} \cdot \bar{B} = \overline{A \cdot B}.$$

2°) Проверить ассоциативность композиции классов: $(\bar{A} \cdot \bar{B}) \cdot \bar{C} = \bar{A} \cdot (\bar{B} \cdot \bar{C})$, исходя из ассоциативности композизии в S, т. е. выбирая в каждом классе *элемент-представитель*.

3°) Пусть E — единичный элемент в S относительно композиции и $\bar{E}$ — класс элемента E. Доказать, что

$$\bar{A} \cdot \bar{E} = \bar{E} \cdot \bar{A} = \bar{A}.$$

(Аналогично п. 2° взять в $\bar{A}$ элемент-представитель.)

5. Пусть имеется алфавит $\{a, b, c\}$ и следующая система соотношений:

$$aa \sim E; \quad bb \sim E; \quad ab \sim ba.$$

Разрешима ли в этом случае проблема эквивалентности?

6. Пусть имеется алфавит $\{a, b, c\}$ и следующая система соотношений:

$$(1)\ aaa \sim E; \quad (2)\ ba \sim ab; \quad (3)\ ca \sim ac.$$

Что можно сказать о словах '*aabaca*' и '*abc*'?

Вообще, разрешима ли *для этого конкретного исчисления* проблема эквивалентности слов?

7. Пусть слова A, B, C, D таковы, что

$$AB = CD \quad \text{и} \quad |A| \leqslant |C|.$$

Доказать, что существует слово X, такое, что

$$C = AX \quad \text{и} \quad B = XD.$$

8. Слово M называется *периодическим*, если оно может быть получено конкатенацией одинаковых слов. Самое короткое слово, повторением которого можно получить M, называется *основанием* слова M.

Рассмотрим непустые слова U и V, такие, что $UV = VU$. Доказать, что U и V являются периодическими и имеют одно и то же основание.

Для решения этой задачи можно воспользоваться следующими указаниями.

Докажите, что для любого целого числа k

$$\underbrace{U \ldots U}_{k \text{ раз}} \cdot \underbrace{V \ldots V}_{k \text{ раз}} = \underbrace{V \ldots V}_{k \text{ раз}} \cdot \underbrace{U \ldots U}_{k \text{ раз}}.$$

Рассмотрите наименьшее общее кратное длины слова U и длины слова V, выберите k и докажите, что U и V являются периодическими. Выяснить, что можно сказать об их основании.

9. 1°) Пусть непустые слова A и B таковы, что для некоторых m и n слова A^m и B^n начинаются одним и тем же сегментом длины $|A| + |B|$. Доказать, что либо $A = B$, либо A и B являются периодическими и имеют одно и то же основание (воспользоваться предыдущим упражнением).

2°) Доказать, что если $A^m = B^n$, $m, n \geqslant 1$, то A и B являются периодическими и имеют одно и то же основание.

3°) Доказать, что если A не пусто, то существует натуральное число k и непериодическое («простое») слово B, такие, что $A = B^k$.

10. Исследовать равенство

$$A\tilde{X} = XB.$$

11. Пусть дан алфавит $\mathfrak{A}$; рассмотрим функцию φ, определенную на множестве слов в алфавите $\mathfrak{A}$ и принимающую значения из этого множества.

Пусть эта функция определяет некоторый эндоморфизм свободной полугруппы; иначе говоря,

$$\varphi(P \cdot Q) = \varphi(P) \cdot \varphi(Q),$$

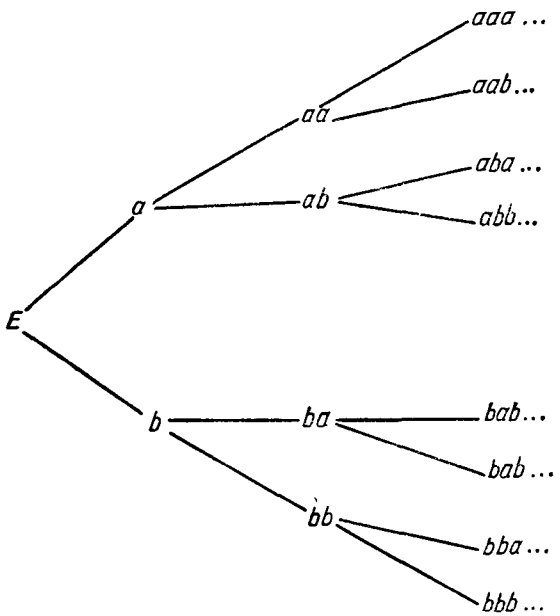
где точкой обозначена конкатенация.

Доказать, что φ вполне определяется теми значениями, которые она принимает на множестве однобуквенных слов; описать несколько простых классов подобных функций.

12. Пусть имеется ассоциативное исчисление, заданное алфавитом $\mathfrak{A} = \{a, b\}$ и следующими соотношениями:

$$aaa \sim E; \quad bbbb \sim E; \quad abb \sim bba; \quad ababa \sim E; \quad babab \sim E.$$

Будем располагать слова этого исчисления в узлах дерева с корнем в E, соблюдая лексикографический порядок, т. е. следующим образом:



Затем будем вычеркивать в этом дереве любое слово, для которого уже записано эквивалентное: например, вычеркнем aaa, так как $aaa \approx E$, затем bba, так как $bba \approx abb$ и т. д.

Найти каноническое представление полученных классов; исследовать факторполугруппу и ядро.

13. Будем рассматривать функции, определенные на свободной полугруппе и принимающие значения в этой полугруппе.

Найти функции α, обладающие свойством

$$\alpha(P \cdot Q) = \alpha[\alpha(P) \cdot \alpha(Q)].$$

Напомним, что пример такой функции был дан в п. 1.2.3.

14. Рассмотрим следующий карточный пасьянс (с колодой из 32 карт): карты выкладываются в ряд; если карта оказывается между двумя картами одинаковой масти или одинакового достоинства, ее можно убрать; если две карты одной и той же масти или достоинства оказываются между картами одинаковой масти или одинакового достоинства, их можно убрать.

Исследовать теорию этого пасьянса.

Глава II

ОБЩИЕ СВЕДЕНИЯ О ФОРМАЛЬНЫХ СИСТЕМАХ

В этой главе мы намереваемся ввести понятие *формальной системы* и проиллюстрировать его, описав с формальной точки зрения некоторый вариант исчисления высказываний. Этот пример можно полностью понять, только имея интуитивное представление об исчислении высказываний; поэтому сначала мы дадим его краткое изложение.

§ 2.1. ОПИСАНИЕ ИСЧИСЛЕНИЯ ВЫСКАЗЫВАНИЙ НА ИНТУИТИВНОМ УРОВНЕ

2.1.1. Высказывания

«Журдэн — персонаж «Мещанина во дворянстве»», «Два плюс два — пять»: эти две фразы в кавычках суть *высказывания*. Первое высказывание истинно, его логическое значение — *истина*; второе высказывание ложно, его логическое значение — *ложь*.

«В этот день шел дождь», «Целое число n является простым» и т. п. — не высказывания, а пропозициональные («высказывательные») переменные, которые в зависимости от обстоятельств могут принимать значения *истина* или *ложь*.

Обычно значение *истина* обозначается единицей, а значение *ложь* — нулем (некоторые авторы используют обратные обозначения).

Таким образом, пропозициональные переменные суть булевы переменные; к этому факту мы еще вернемся.

Исчисление высказываний не занимается *анализом* высказываний, а рассматривает их как неразложимые единицы — с точки зрения возможностей по-разному комбинировать их с помощью различных логических операций, к обзору которых мы и перейдем.

Высказывания и пропозициональные переменные мы будем обозначать малыми латинскими буквами p, q, ...

2.1.2. Конъюнкция

Высказывание «Идет дождь и дует ветер» считается истинным тогда и только тогда, когда одновременно идет дождь и дует ветер. Это высказывание является *конъюнкцией* высказываний «Идет дождь» и «Дует ветер».

Более точно, конъюнкция двух высказываний (или пропозициональных переменных) p и q есть высказывание (пропозицио-

нальная переменная), обозначаемое $p \wedge q$ и имеющее в четырех возможных случаях распределения значений p и q следующие логические значения:

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Помимо символа $\wedge$ конъюнкцию обозначают также символами & и · или простым соположением высказываний.

2.1.3. *Дизъюнкция*

Аналогичным образом вводится понятие *дизъюнкции.* Дизъюнкция определяется следующей таблицей:

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

Дизъюнкция $p \vee q$ считается истинной тогда и только тогда, когда истинно хотя бы одно из высказываний p или q. Дизъюнкция соответствует неразделительному употреблению русского «или» в такой фразе, как «Он дурак или мерзавец» (подразумевается: «А может быть, и то, и другое»).

Дизъюнкцию обозначают иногда также символом $+$.

2.1.4. *Отрицание*

Любому высказыванию (пропозициональной переменной) p можно поставить в соответствие высказывание $\neg p$ — его *отрицание*, которое определяется следующей таблицей:

p	$\neg p$
0	1
1	0

Другие обозначения отрицания: $\bar{p}$, $\sim p$, $\leftharpoondown p$.

2.1.5. *Импликация*

Любой паре высказываний p, q можно поставить в соответствие высказывание $p \supset q$, определяемое следующей таблицей:

p	q	$p \supset q$
0	0	1
0	1	1
1	0	0
1	1	1

Напоминаем еще раз, что высказывания p и q не анализируются, их содержание не берется в расчет ни в какой степени, и единственное, что о них известно, — это их логические значения. Поэтому *импликация*, которая была только что определена, не имеет никакого отношения к причинно-следственной зависимости или к выводу q из p (к доказательству q на основе p). Логический оператор $\supset$ в выражении «$p \supset q$» ни в коем случае нельзя смешивать с метаимпликацией «$p \Rightarrow q$» (означающей, что q действительно выводимо из p).

Примеры.

p : Наполеон умер на острове Св. Елены.
q : Верцингеторикс носил усы.
$p \supset q$ принимает значение q.

p : $2 \times 2 = 5$.
q : 12 — простое число.
$p \supset q$ принимает значение 1.

p : Луна состоит из овечьего сыра.
q : 17 — простое число.
$p \supset q$ принимает значение 1.

p : 17 — простое число.
q : 16 — простое число.
$p \supset q$ принимает значение 0.

Здесь нет никакого парадокса; вскоре мы в этом убедимся.

Правило вывода (modus ponens). Если мы знаем (откуда — нас сейчас не касается!), что p имеет значение 1 и $p \supset q$ имеет значение 1, то мы можем сделать вывод, что и q имеет значение 1.

2.1.6. *Логическая эквивалентность*

Из двух высказываний (или пропозициональных переменных) p и q можно образовать высказывание $p \equiv q$, определяемое таблицей

p	q	$p \equiv q$
0	0	1
0	1	0
1	0	0
1	1	1

Здесь фактически представлено высказывание «p эквивалентно q», или, что то же самое, «p имеет то же логическое значение, что и q».

Как и в случае импликации, при образовании *логической эквивалентности* реальное содержание соединяемых высказываний совершенно не принимается во внимание.

2.1.7. Классификация логических выражений

Пусть у нас имеется конечное или бесконечное число пропозициональных переменных $p, q, \ldots$; будем образовывать из них выражения последовательным применением логических операций $\wedge$, $\vee$, $\neg$, $\supset$, $\equiv$.

Примеры.

$$(p \vee p) \wedge q, \quad (p \supset q) \vee (q \supset p).$$

1) Существуют выражения, которые принимают значение 1, каковы бы ни были значения входящих в них переменных.

Примеры.

$$p \vee \neg p, \qquad p \supset p.$$

В силу якобы парадоксального характера нашей импликации таким является и выражение

$$p \supset (q \supset p) \ldots . \qquad \text{NB!}$$

Такие выражения называются *тождественно истинными*, или *тавтологиями*.

2) Существуют выражения, которые принимают значение 0, каковы бы ни были значения входящих в них переменных.

Примеры.

$$p \wedge \neg p, \quad (p \vee \neg p) \supset (p \wedge \neg p).$$

Такие выражения называются *тождественно ложными*, или *противоречивыми*.

3) Существуют, наконец, выражения, которые принимают то значение 1, то значение 0 в зависимости от того, каковы значения входящих в них переменных.

Пример. $(p \supset q) \vee (p \wedge q)$.
В самом деле, обозначив это высказывание через r, мы получим

p	q	r
0	0	1
0	1	1
1	0	0
1	1	1

Поиск тавтологий представляет особый интерес, поскольку в логике тавтологии играют ту же роль, что в алгебре — тождества.

Пример. Выражение

$$(\neg q \supset \neg p) \supset (p \supset q)$$

является тавтологией (что нетрудно доказать, построив таблицу истинности). Эта тавтология лежит в основе *правила контрапозиции*.

Тавтологическая эквивалентность. Мы будем писать

$$r = s,$$

желая выразить тот факт, что высказывание

$$r \equiv s$$

является тавтологией.

Примеры.

$$(p \vee q) \vee m = p \vee (q \vee m),$$
$$(p \wedge q) \wedge m = p \wedge (q \wedge m),$$
$$p \supset q = \neg p \vee q), \qquad \text{NB!}$$
$$p \equiv q = (p \supset q) \wedge (q \supset p).$$

Отношение $=$ (мы будем называть его *тавтологической эквивалентностью*) означает, что при любых условиях выражения, стоящие слева и справа от символа $=$, принимают одно и то же логическое значение. Это отношение является отношением эквивалентности (иначе говоря, оно рефлексивно, симметрично и транзитивно).

Константа *истина* тавтологически эквивалентна любой тавтологии. Например:

$$p \vee \neg p = \textit{истина}.$$

Константа *ложь* тавтологически эквивалентна любому противоречивому выражению. Например:

$$p \wedge \neg p = \textit{ложь}.$$

2.1.8. Правила де Моргана

Легко убедиться в том, что выражения

$$\neg(p \vee q) = \neg p \wedge \neg q,$$
$$\neg(p \wedge q) = \neg p \vee \neg q$$

являются тавтологическими эквивалентностями. Эти эквивалентности называются правилами де Моргана.

Для большей компактности записи будем писать $\bar{p}$ вместо $\neg p$. Для любых переменных или выражений r, s, t имеем

1°. $r \vee r = r$; $r \wedge r = r$: идемпотентность;

2°. $r \vee s = s \vee r$; $r \wedge s = s \wedge r$: коммутативность;

3°. $r \vee (s \vee t) = (r \vee s) \vee t$; $r \wedge (s \wedge t) = (r \wedge s) \wedge t$: ассоциативность;

4°. $r \vee (r \wedge s) = r = r \wedge (r \vee s)$;

5°. $\left.\begin{array}{l} r \vee (s \wedge t) = (r \vee s) \wedge (r \vee t) \\ r \wedge (s \vee t) = (r \wedge s) \vee (r \wedge t) \end{array}\right\}$: дистрибутивность;

6°. $r \vee \bar{r} = \textit{истина}$; $r \wedge \bar{r} = \textit{ложь}$;

7°. $\left.\begin{array}{l} \overline{r \vee s} = \bar{r} \wedge \bar{s} \\ \overline{r \wedge s} = \bar{r} \vee \bar{s} \end{array}\right\}$: двойственность (правила де Моргана);

8°. $\overline{(\bar{r})} = r$: инволютивность отрицания.

Соотношения 1°—7° показывают, что исчисление высказываний снабжает множество логических выражений структурой булевой алгебры — это следует из самого определения булевой алгебры. (Точнее, мы имеем здесь в виду выражения, полученные из элементов счетного множества переменных $p, q \ldots$ путем применения унарной операции $\neg$ и бинарных операций $\vee$ и $\wedge$ произвольное число раз, причем тавтологически эквивалентные выражения считаются равными.)

2.1.9. Теоретико-множественная интерпретация исчисления высказываний

Возьмем семейство множеств P, Q ... и будем интерпретировать пропозициональные переменные p, q, ... как выражения вида

$$x \in P,\ x \in Q,\ \ldots \text{ с одним и тем же } x.$$

Тогда $p \wedge q$ интерпретируется как $x \in P \cap Q$, $p \vee q$ — как $x \in P \cup Q$ и т. п.

Исчисление высказываний получает теперь следующую интерпретацию. Его теоретико-множественным образом является алгебра множеств, получаемых из P, Q, ... с помощью операций пересечения, объединения и взятия дополнения относительно некоторого универсального множества U, содержащего все P, Q, ... и

соответствующего *истине*. Взятие дополнения относительно U соответствует отрицанию; пустое множество $\emptyset$, являющееся дополнением для U, соответствует *лжи*.

2.1.10. Варианты исчисления высказываний

Можно строить исчисления, эквивалентные описанному выше, но с другими основными операциями.

Один из наиболее интересных вариантов исчисления высказываний получается в случае, когда используются только операции $\neg$ и $\supset$. Мы знаем, что

$$p \supset q = \neg p \vee q$$

и что

$$\neg \neg p = p,$$

откуда

$$p \vee q = \neg p \supset q.$$

Выражение для $p \wedge q$ можно получить с помощью правила де Моргана.

Подчеркнем, что все операции исчисления высказываний могут быть выражены через одну подходящим образом выбранную операцию (см. ниже упр. 2.1.11.6).

2.1.11. Упражнения

1. Пусть даны высказывания: 'жарко $\equiv c$', 'идет дождь $\equiv p$', 'воздух сухой $\equiv s$'; изобразить с помощью символов следующие высказывания:

а) 'идет дождь и жарко',

б) 'жарко, но воздух не сухой',

в) 'воздух сырой или жарко',

г) 'дождь не идет или воздух сухой'.

2. Предположив, что предложение «Холодно и воздух сырой» истинно, исследовать истинностные значения предыдущих предложений.

3. Обозначим через p предложение «Самолет летит высоко» и через q — «Самолет набирает высоту». Перевести на русский язык:

а) $p \wedge q$,

б) $p \wedge q$,

в) $\bar{p} \wedge \bar{q}$,

г) $p \vee \bar{q}$,

д) $\overline{p \wedge q}$,

е) $\overline{p \vee q}$,

ж) $\overline{\bar{p} \vee \bar{q}}$.

Попытайтесь выразить полученные переводы с помощью более простых символических выражений.

4. Пусть p означает «У меня есть проигрыватель», а q — «У меня есть пластинки». Переведите на русский язык, а затем упростите выражение

$$\overline{\bar{p} \vee \bar{q}} \wedge \bar{p}.$$

5. Исходя из двух переменных или выражений, можно построить формулу «pWq», где W задается следующей таблицей:

p	q	pWq
0	0	0
0	1	1
1	0	1
1	1	0

W соответствует разделительному *или* (т. е. строгой дизъюнкции: либо одно, либо другое, но не то и другое вместе).

Дайте теоретико-множественную интерпретацию операции W. Является ли эта операция ассоциативной?

6. Аналогичный вопрос для операции $|$, которая называется «штрихом Шеффера» (или *несовместимостью*) и определяется следующей таблицей:

p	q	$p \mid q$
0	0	1
0	1	1
1	0	1
1	1	0

Проверить следующие тавтологические эквивалентности:

$$\neg p = p \mid p,$$
$$p \vee q = (p \mid p) \mid (q \mid q),$$
$$p \wedge q = (p \mid q) \mid (p \mid q),$$
$$p \supset q = p \mid (q \mid q).$$

§ 2.2. ПОНЯТИЕ ФОРМАЛЬНОЙ СИСТЕМЫ

Исторически понятие формальной системы развилось на основе понятия аксиоматической теории, о котором поэтому мы скажем здесь несколько слов.

Частично аксиоматизированным было уже изложение геометрии в «Началах» Евклида, однако полностью аксиоматический подход четко оформился лишь в конце 19 века.

2.2.1. Аксиоматическая теория

Внутренняя логическая связность некоторой математической теории и ее адекватность для описания того или иного круга физических явлений — это две совершенно разные вещи. Данный факт был осознан математиками, когда им пришлось объяснять возможность одновременного существования евклидовой геометрии и различных неевклидовых геометрий.

Чтобы построить и логически развивать некоторую математическую теорию, необходимо абстрагироваться от природы объектов, которые она рассматривает (вернее, от природы объектов, которые, так сказать, мотивировали извне создание этой теории). В то же время необходимо самым тщательным образом уточнить с помощью *аксиом* основные отношения, связывающие эти объекты друг с другом.

«Мы мыслим три различные системы вещей: вещи первой системы мы называем точками; вещи второй системы мы называем прямыми; вещи третьей системы мы называем плоскостями», — писал Давид Гильберт в своих «Основаниях геометрии»; в устных беседах он добавлял: «Конечно, эти вещи можно было бы называть не точками, прямыми и плоскостями, а, скажем, столами, стульями и пивными кружками».

Вся существенная информация о точках и прямых содержится в следующих аксиомах:

1.1. Прямая есть множество точек.

1.2. Через любые две точки можно провести некоторую прямую.

1.3. Через две точки можно провести только одну прямую.

1.4. На любой прямой имеются по крайней мере две точки.

1.5. Существуют по крайней мере три точки, не принадлежащие одной прямой.

Исходя из аксиом, можно доказывать теоремы, например:

Теорема. *Любые две различные прямые могут иметь только одну общую точку.*

Каким именно способом выводятся теоремы из аксиом в рамках аксиоматической теории, об этом в большинстве случаев специально не говорят. Считается, что это делается в соответствии с правилами логики, которые основаны на здравом смысле и усваиваются на примерах. Эти правила к тому же представляются «очевидными», например: если некоторое высказывание истинно, то его отрицание ложно.

Долгое время, по крайней мере во Франции, исследование логических правил было предоставлено «логикам», которые обыч-

но не знали математики (и работы которых считались бесперспективными).

Эта ситуация изменилась, как только математики стали заниматься математическими объектами, столь малоинтуитивными по своему характеру, что для их понимания «логики здравого смысла» оказалось недостаточно.

Переход от «наивных аксиоматических теорий» к «формальным системам» связан с осознанием того факта, что само понятие «логических правил» содержит в себе нечто неясное и потому нуждается в уточнении. Но тогда исследователь оказывается в ситуации, которая аналогична ситуации на уровне аксиом: мы по своему произволу выбирали аксиому Евклида или аксиому Лобачевского; точно так же мы можем по своему произволу выбирать одни или другие правила вывода и получать тем самым разные формальные системы.

2.2.2. Метаязык

Формальная система имеет дело с языком L, который строится над некоторым алфавитом $\mathfrak{A}$ с помощью аксиом и правил вывода. В свое время мы уточним это, а пока нам достаточно заметить следующее. Чтобы говорить об элементах формальной системы, о ее правилах и т. д., приходится пользоваться каким-то языком; этот язык должен содержать имена объектов, имена отношений и т. п. В данном переводе настоящей книги для этой цели применяется обычный русский язык, дополненный некоторыми символами, отличными от символов самой формальной системы.

Пример. Рассмотрим формальную систему, определенную над алфавитом $\mathfrak{A} = \{a, b, c\}$; в ней будет использоваться конкатенация.

Чтобы говорить о словах этой формальной системы, мы будем прибегать к таким именам, как

$$`abaca',$$

образованным из букв алфавита с помощью одинарных кавычек, или к таким обозначениям, как A, B и т. д.

Заметим, что специальные (одинарные) кавычки будут использоваться для выделения упоминаний о словах формальной системы, а обычные (двойные) кавычки — для выделения упоминаний о символах, используемых для обозначения слов формальной системы.

Для обозначения операции конкатенации будем применять символ «$\cdot$» и писать, например, так:

Если $A =$ '*abaca*' и если $B =$ '*ba*',

то $A \cdot B =$ '*abacaba*'.

Знак «=» обозначает здесь, что выражение, стоящее слева от него, и выражение, стоящее справа от него, — это два *разных имени одного и того же объекта.*

Сказанное выше уже, видимо, позволяет читателю получить некоторое предварительное представление о *метаязыке.* Одни авторы обозначают этим термином определенную систему знаков, для описания которой требуется в свою очередь некоторый метаметаязык. Другие называют метаязыком объединение обычного естественного языка и набора специальных знаков, позволяющее описать формальную систему. Мы будем в основном придерживаться второй точки зрения, чтобы избежать иерархии метаязыков, ненужной для наших целей.

Метаязыковые символы, употребляемые в дальнейшем, будут определяться с помощью по возможности однозначных и логически отчетливых русских фраз. Таким образом, эти символы по существу представляют собой сокращения.

Пример. Пусть алфавит формальной системы есть

$$\mathfrak{A} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -\}.$$

Мы хотим определить два множества слов, а именно множество «целых чисел без знака» и множество «целых чисел». Чтобы сформулировать правила образования этих множеств, мы введем метаязыковую переменную C, которая принимает любое значение из множества $\{0, 1, \ldots, 9\}$, метаязыковую переменную ESS[1]), принимающую значение из множества «целых чисел без знака», и метаязыковую переменную E, принимающую значение из множества «целых чисел».

Ломаные скобки «⟨» и «⟩» будут использоваться для разграничения вхождений переменных; вертикальная черточка «|» обозначает союз *или* в одном из его значений (читатель сам уточнит, в каком именно); наконец, символ «::=» означает разрешение заменить то, что стоит слева от него, каким-либо из выражений, стоящих справа от него и разделенных вертикальными черточками. Для обозначения конкатенации используется «классический» типографский прием — простое соположение (без специального знака).

Приведем правила порождения интересующих нас множеств.

Правило 1: $\langle ESS\rangle ::= \langle C\rangle \mid \langle ESS\rangle\langle C\rangle$.

Правило 2: $\langle E\rangle ::= \langle ESS\rangle \mid +\langle ESS\rangle \mid -\langle ESS\rangle$.

Применяя правило 1, мы можем получить, например,

$$\langle ESS\rangle ::= 3,$$
$$\langle ESS\rangle ::= 35;$$

[1]) ESS = Entier Sans Signe. — *Прим. перев.*

применение правила 2 к последнему результату дает, например,

$$\langle E\rangle :: = -35.$$

Предположим, что алфавит некоторой формальной системы содержит букву «$\wedge$» и имеется правило: «Если A и B — слова типа θ, то слово $A \cdot \wedge \cdot B$ — тоже слово типа θ». Русское выражение «если..., то ..» представляется в данном контексте абсолютно однозначным; однако сугубо для краткости записи его можно заменить *метаязыковым символом*, например символом «$\Rightarrow$».

В этом месте читатель может быть удивлен следующим обстоятельством. «Разоблачив» нечеткость и недостаточность «логики здравого смысла», мы тем не менее включаем в наш метаязык такие выражения, как *«если..., то...»* (замаскировав его обозначением $\Rightarrow$), «разделительное *или*» и т. д. На это мы ответим, что для общения, для передачи мыслей использовать какой-нибудь «язык» необходимо; но мы ограничиваем употребление метаязыка исключительно описанием правил, т. е. мы используем «логику здравого смысла» не для самих логических выводов, а только для описания их структуры, для того, чтобы понять «правила игры».

Кроме того, мы не собираемся *обосновывать* какую-либо математическую теорию (в философском смысле термина «обосновывать»). Наша цель состоит только в том, чтобы придать максимальную логическую отчетливость каждому нашему шагу. Мы располагаем объективным критерием, позволяющим установить, достигнута ли эта скромная цель: если да, то любые логические машины, созданные специально, чтобы применять правила, будут всегда давать в одних и тех же условиях одни и те же результаты.

2.2.3. Синтаксис и значение

Ясно, что, вообще говоря, ученые не строят формальные системы из чисто эстетических соображений или только для собственного удовольствия. Обычно формальная система создается как идеализированный образ чего-то другого — это «другое» можно было бы назвать, например, «теорией», — по отношению к чему эта формальная система и обретает смысл.

Тогда формальная система представляет собой формализацию некоторой теории, образуя *синтаксис* последней; теория же представляет собой *интерпретацию* данной формальной системы. Правильно построенные слова (правильные фразы) формальной системы получают смысл в рамках формализуемой теории; обратно, любому высказыванию, имеющему смысл в данной теории, соответствует некоторый класс правильно построенных слов (правильных фраз) в формальной системе.

Вопрос об адекватности данной формальной системы для данной теории — это проблема, относящаяся к методологии науки.

Исследование подобных проблем должно, разумеется, вестись как можно более строгими методами, но оно может выходить за рамки математики как таковой.

Мы попытаемся проиллюстрировать все эти общие соображения, снова обратившись к исчислению высказываний, но теперь рассматривая его на гораздо более формальном уровне.

§ 2.3. ФОРМАЛИЗОВАННЫЙ ВАРИАНТ ИСЧИСЛЕНИЯ ВЫСКАЗЫВАНИЙ

2.3.1. Алфавит

Пусть имеется счетное множество абстрактных математических объектов, образующих абстрактный алфавит. Эти объекты находятся во взаимно однозначном соответствии со следующими графическими символами, образующими алфавит:

$$\mathfrak{A} = \{\,], [, \neg, \supset, a_1, \ldots, a_n, \ldots\}.$$

Пояснения:

«{», «}» и «,» суть символы метаязыка;

«]» и «[» суть имена формальных скобок;

«$\neg$» есть имя формального отрицания;

«$\supset$» есть имя формальной импликации;

a_i — имена элементарных объектов, называемых *атомами* и образующих счетное множество.

2.3.2. Формулы

Мы выделим в полугруппе $\mathfrak{A}^*$ некоторую часть — формальный язык, соответствующий некоторому множеству конкретных формул (служащих представлениями абстрактных формул).

Правила, задающие это множество формул, предписывают определенное «правильное» употребление скобок и символов операций. Вот эти правила:

F1. Всякий атом есть формула.

F2. Если слово X — формула, то слово $\neg X$ — тоже формула.

F3. Если слова X и Y — формулы, то слово $[X \supset Y]$ — тоже формула.

F4. Никаких других формул нет.

X и Y являются здесь метаязыковыми переменными

Примеры.

a_1 является формулой в соответствии с F1;

a_2 является формулой в соответствии с F1;

$[a_1 \supset a_2]$ является формулой в соответствии с F3;

$\neg [a_1 \supset a_2]$ является формулой в соответствии с F2;

a_3 является формулой в соответствии с F1;

$[\neg [a_1 \supset a_2] \supset a_3]$ является формулой в соответствии с F3 и предыдущим замечанием.

Множество всех формул можно получить, последовательно строя формулы все большей длины.

Формулы длины 1 — это атомы. Поскольку в формулах ничего не стирается, каждая формула содержит по крайней мере один атом.

Формулы длины 2 таковы:

$$\neg a_1,\ \neg a_2,\ \ldots,\ \neg a_n,\ \ldots,$$

затем идут формулы длины 3:

$$\neg\neg a_1,\ \neg\neg a_2,\ \ldots,\ \neg\neg a_n,\ \ldots,$$

формулы длины 4:

$$\neg\neg\neg a_1,\ \neg\neg\neg a_2,\ \ldots,\ \neg\neg\neg a_n,\ \ldots,$$

формулы длины 5:

$$\neg\neg\neg\neg a_1,\ \neg\neg\neg\neg a_2,\ \ldots,\ \neg\neg\neg\neg a_n,\ \ldots$$

и

$$[a_1 \supset a_1],\ [a_1 \supset a_2],\ \ldots,\ [a_n \supset a_p],\ \ldots \text{ и т. д.}$$

Если дано слово $M \in \mathfrak{A}^*$, то всегда можно решить, является ли M формулой.

Если M не содержит атомов, то M не является формулой.

Если M содержит атомы, то нужно построить из этих атомов ьсе возможные формулы длины, меньшей или равной длине M. Слово M является формулой тогда и только тогда, когда M является одной из этих формул. Ясно, что указанный процесс определения того, является ли M формулой, может быть «механизирован». При этом для нас здесь неважно, насколько такой процесс «удобен» или «эффективен»; этот вопрос обсуждаться не будет.

Примеры.

$\neg$[не является формулой (в данном выражении нет атомов).

$[\neg a_2 \supset a_2]$ является формулой, поскольку это выражение содержится в множестве всех формул длины $\leqslant 6$, которые можно построить из a_2; это множество состоит из следующих формул:

a_2,
$\neg a_2$,
$\neg\neg a_2$,
$\neg\neg\neg a_2$,
$\neg\neg\neg\neg a_2$, $[a_2 \supset a_2]$,
$\neg\neg\neg\neg\neg a_2$, $\neg[a_2 \supset a_2]$, $[\neg a_2 \supset a_2]$,
$[a_2 \supset \neg a_2]$.

2.3.3. Множество элементарных теорем

Теперь мы рассмотрим другой (формальный) язык, содержащийся в предыдущем; этот язык мы назовем множеством *теорем*

Теоремы соответствуют тому, что в неформальном варианте называлось тавтологиями.

Сначала мы определим *элементарные теоремы* с помощью следующих соглашений:

Если слова X, Y, Z — формулы, то

T1. $[X \supset [Y \supset X]]$ есть элементарная теорема.

T2. $[[\neg X \supset \neg Y] \supset [Y \supset X]$ есть элементарная теорема.

T3. $[[X \supset [Y \supset Z]] \supset [[X \supset Y] \supset [X \supset Z]]]$ есть элементарная теорема.

T4. Других элементарных теорем нет.

Если рассматривать элементарные теоремы как *аксиомы*, то выражения T1—T3, принадлежащие метаязыку, будут *схемами аксиом*.

Легко убедиться в том, что элементарные теоремы являются формулами в силу F2 и F3. Нетрудно показать также, что для любого слова M можно решить, является это слово элементарной теоремой или нет.

2.3.4. О выборе элементарных теорем

Отметим, что T1 отражает «квазипарадоксальный» характер импликации. T2 есть правило контрапозиции. Необходимо особо обратить внимание на следующий факт: когда мы используем содержательное понятие истины, ясно, что

$$\neg(\neg p) = p;$$

поэтому в этом случае безразлично, писать

$$[\neg X \supset \neg Y] \supset [Y \supset X]$$

или

$$[Y \supset X] \supset [\neg X \supset \neg Y].$$

Однако дело обстоит совсем не так, когда мы с самого начала не прибегаем к понятию истины (т. е. когда мы имеем в виду другую интерпретацию).

Выбор T3 обусловлен техническим удобством.

Исходя из элементарных теорем, мы будем строить теоремы, задав a priori правило вывода.

2.3.5. Правило вывода

Если слова X и $[X \supset Y]$ суть теоремы, то слово Y — тоже теорема.

Легко видеть, что теоремы обязательно являются формулами.

Заметим, что введенное нами правило есть известное правило modus ponens.

2.3.6. Примеры вывода

В качестве примера и одновременно упражнения мы покажем, что $[a_1 \supset a_1]$ и вообще $[X \supset X]$ суть теоремы.

Для этого мы построим теорему, в которой $[a_1 \supset a_1]$ окажется в позиции Y из правила 2.3.5.

(1) a_1 является формулой в силу F1.

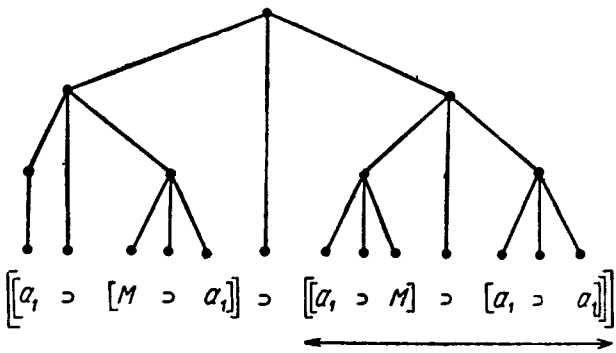
(2) Слово $[a_1 \supset a_1]$, которое мы будем обозначать впредь через M, является формулой в силу (1) и F3.

(3) Слово $[a_1 \supset M]$ является формулой в силу (1), (2) и F3.

(4) Слово $[a_1 \supset M]$ является теоремой в силу (1) и T1.

(5) Слово $[a_1 \supset [M \supset a_1]]$ является теоремой в силу (1), (2) и T1.

(6) Слово $[[a_1 \supset [M \supset a_1]] \supset [[a_1 \supset M] \supset [a_1 \supset a_1]]]$, структура которого может быть изображена деревом



является теоремой в силу (1), (2) и T3.

(7) Правая часть слова (6), под которой на рисунке помещена стрелка, является теоремой в силу (5), (6) и правила modus ponens.

(8) $[a_1 \supset a_1]$ является теоремой в силу (4), (7) и правила modus ponens.

Точно так же можно доказать, что для любой формулы X выражение $[X \supset X]$ является теоремой.

2.3.7. Заключение

Итак, соблюдая некоторые «правила игры», мы построили «систему», состоящую из двух связанных частей. Мы можем придать этой системе определенный смысл: она формализует «исчисление высказываний, рассматриваемое на интуитивном уровне понимания»; стало быть, она формализует в какой-то степени «логику здравого смысла».

Заметим, что построенная система не зависит от интерпретации, которую мы ей даем; возможны и другие интерпретации.

Теперь остается показать, каким образом в рамках данной формальной системы можно было бы, действуя вполне строго, ввести истинностные значения; однако мы этого делать не будем.

§ 2.4. ОПРЕДЕЛЕНИЕ ФОРМАЛЬНОЙ СИСТЕМЫ

2.4.1. *Формальные системы*

Резюмируем сказанное выше о формальных системах. Мы будем называть *логистической системой*[1]) упорядоченную четверку множеств:

$$\mathfrak{S} = \langle \mathfrak{X}, \mathfrak{F}, \mathfrak{A}, \mathbf{R} \rangle,$$

где $\mathfrak{X}$ — счетный алфавит, буквы которого могут считаться перенумерованными;

$\mathfrak{F}$ — (формальный) язык над $\mathfrak{X}$, или *множество формул* ($\mathfrak{F} \subset \mathfrak{X}^*$);

$\mathfrak{A}$ — некоторое подмножество множества $\mathfrak{F}$, или *множество аксиом,* задаваемое с помощью схем аксиом;

$\mathbf{R}$ — множество правил вывода (их вид уточняется ниже).

2.4.2. *Правила вывода*

Рассмотрим декартово произведение $\mathfrak{F} \times \ldots \times \mathfrak{F}$, состоящее из n сомножителей ($n \geqslant 1$); его элементами являются упорядоченные n-ки формул.

Правило вывода $\mathfrak{R}$ есть правило, позволяющее поставить в соответствие каждой n-ке $\langle y_1, y_2, \ldots, y_n \rangle$ из некоторого подмножества произведения $\mathfrak{F} \times \ldots \times \mathfrak{F}$ одну определенную формулу x.

Более точно, правило $\mathfrak{R}$ определяется некоторым подмножеством декартова произведения $\mathfrak{F}^n \times \mathfrak{F}$, причем это подмножество должно обладать следующим свойством: ни для какой n-ки $\langle y_1, y_2, \ldots, y_n \rangle \in \mathfrak{F}^n$ не может существовать более одной формулы $x \in \mathfrak{F}$, для которой $\langle \langle y_1, y_2, \ldots, y_n \rangle, x \rangle$ принадлежит данному подмножеству. *Областью определения* правила $\mathfrak{R}$ является некоторое подмножество $\mathfrak{F}^n$.

При конкретном применении правила вывода формулы $y_1, \ldots, y_n$ называют *посылками*, а формулу x — *заключением*; посылки и заключение вместе суть *аргументы* примененного правила.

2.4.3. *Комбинаторные системы*

Если алфавит $\mathfrak{X}$ конечен, $\mathfrak{F}$ совпадает с $\mathfrak{X}^*$ и имеется ровно одна аксиома, то $\mathfrak{S}$ представляет собой *комбинаторную систему*; комбинаторные системы будут рассмотрены ниже (см. гл. III).

---

[1]) Термин заимствован у Ж. Порта (Porte J., Recherches sur la théorie générale des systèmes formels et sur les systèmes connectifs, Paris, Gauthier-Villars, 1965).

2.4.4. Выводимость

Пусть $\Phi = \langle f_1, f_2, \ldots, f_k \rangle$ — упорядоченное множество (последовательность) формул.

Если существуют правило $\mathfrak{R}$ и формула x, такие, что $\Phi \mathfrak{R} x$, то x называется *непосредственным следствием* из Φ.

Множество правил вывода $\mathbf{R}$ определяет, таким образом, отношение *непосредственной выводимости*: $\Phi \mathbf{R} x$ тогда и только тогда, когда существует правило вывода $\mathfrak{R}$, такое, что $\Phi \mathfrak{R} x$. Отношение $\mathbf{R}$ определяется в декартовом произведении множества последовательностей формул на множество формул.

Пусть $\mathbf{M}$ — произвольное множество формул. Обозначим через E_0 объединение $\mathbf{M}$ и множества аксиом. Образуем всевозможные конечные последовательности элементов E_0 и добавим к E_0 все формулы, являющиеся непосредственными следствиями из таких последовательностей. Полученное множество обозначим через E_1. Построим, далее, множество E_2 с помощью E_1 точно так же, как E_1 строилось с помощью E_0, и т. д. Формулы, принадлежащие $E_1, E_2, \ldots, E_n, \ldots$, будем называть *выводимыми* из $\mathbf{M}$, или *следствиями* из $\mathbf{M}$. Выделяя формулы, действительно участвующие в получении выводимой из $\mathbf{M}$ формулы y, получим *вывод* этой формулы из $\mathbf{M}$. (Точнее, вывод есть дерево, строящееся следующим образом: корнем его служит y; в концах дуг, исходящих из корня, стоят формулы, непосредственным следствием которых является y, и т. д.; висячие вершины дерева будут элементами E_0.) Запись $\mathbf{M} \underset{\mathfrak{S}}{\models} y$ означает, что «y выводима из $\mathbf{M}$ в $\mathfrak{S}$».

Замечания. а) $\mathbf{M}$ определяется только составом своих элементов; их порядок значения не имеет. Множество $\mathbf{M}$ называется множеством *посылок*.

б) Если множество $\mathbf{N}$ содержит все элементы из $\mathbf{M}$ ($\mathbf{M} \subset \mathbf{N}$) и $\mathbf{M} \underset{\mathfrak{S}}{\models} y$, то ясно, что $\mathbf{N} \underset{\mathfrak{S}}{\models} y$; иначе говоря, к посылкам можно присоединять любые другие.

в) Из непосредственной выводимости следует выводимость.

г) Всякая формула выводима из самой себя.

2.4.5. Теоремы

Формулы, выводимые из аксиом некоторой формальной системы, называются ее *теоремами*.

Характеристическое свойство теорем состоит в том, что они выводимы из любого множества посылок, в том числе — из пустого множества.

Вместо $\emptyset \underset{\mathfrak{S}}{\models} y$ мы будем писать сокращенно $\underset{\mathfrak{S}}{\models} y$.

2.4.6. Упражнения

1. Доказать, что из непосредственной выводимости следует выводимость и что всякая формула выводима из самой себя.

2. Определим формальную систему, имеющую тот же алфавит, то же множество формул и то же правило вывода, что и система, введенная выше (стр. 42—44), для описания исчисления высказываний.

Схемы аксиом этой системы таковы:

T1′. $[X \supset [Y \supset X]]$.

T2′. $[[X \supset [Y \supset Z]] \supset [[X \supset Y] \supset [X \supset Z]]]$.

T3′. $[\neg X \supset [X \supset Y]]$.

Выяснить, являются ли следующие формулы теоремами:

(1) $[[Y \supset Z] \supset [[X \supset Y] \supset [X \supset Z]]]$;

(2) $[X \supset [[X \supset Y] \supset Y]]$;

(3) $[[X \supset Y] \supset [[Y \supset Z] \supset [X \supset Z]]]$.

3. Можно ли в рамках системы предыдущего упражнения вывести $[Y \supset [X \supset Z]]$ из $[X \supset [Y \supset Z]]$?

4. В п. 2.3.1 мы рассматривали *бесконечный* алфавит. Показать, что определение можно изменить таким образом, чтобы алфавит был конечным.

5. В п. 2.2.1 были приведены некоторые аксиомы, касающиеся точек и прямых.

Опираясь на *семантические* соображения, т. е. привлекая *содержательную интерпретацию* формальной системы, показать, что аксиома «через две точки можно провести только одну прямую» не следует из аксиомы «через любые две точки можно провести некоторую прямую».

Для этого можно, например, заменить слово «точка» словом «штучка», а слово «прямая» — словом «вещь». Затем надо подобрать такую интерпретацию штучки и вещи, для которой вторая аксиома выполняется, а первая — нет.

Глава III

КОМБИНАТОРНЫЕ СИСТЕМЫ

Комбинаторные системы — это формальные системы особого типа. Как показывает само их название, они используются при исследовании задач *комбинаторного* характера (а именно, относящихся к словам).

Введем сначала понятия, необходимые для формулирования правил вывода в комбинаторной системе.

Будем рассматривать свободную полугруппу, определенную над алфавитом

$$\mathfrak{A} = \{a_i \mid 1 \leqslant i \leqslant n\};$$

в примерах буквы этого алфавита обозначаются строчными латинскими буквами.

Напомним, что через E обозначается пустое слово, а через $|A|$ — длина слова A.

§ 3.1. ОПРЕДЕЛЕНИЕ КОМБИНАТОРНЫХ СИСТЕМ

3.1.1. Продукции

Когда мы говорили об исчислении слов, мы рассматривали двусторонние соотношения вроде '*apa*' $\sim$ '*ло*'.

В настоящей главе мы встанем на несколько иную точку зрения, а именно введем в наши правила *направление*.

Попытаемся построить как можно более общее определение, из которого в дальнейшем будет получено много полезных определений более частного вида. Пусть имеется упорядоченная шестерка слов $\langle G, H, K; \overline{G}, \overline{H}, \overline{K}\rangle$, записанных в алфавите $\mathfrak{A}$. Рассмотрим слово X вида

$$X = GPHQK,$$

т. е. слово, начинающееся вхождением слова G, за которым следует вхождение H, отделенное от G вхождением некоторого слова P (возможно, пустого) и оканчивающееся вхождением слова K, отделенным от H вхождением слова Q (которое также может быть пустым).

Замещая вхождения G, H и K вхождениями слов $\overline{G}$, $\overline{H}$ и $\overline{K}$ соответственно, мы получаем слово $Y = \overline{G}P\overline{H}Q\overline{K}$.

Пример. Пусть $\mathfrak{A}$ — латинский алфавит. Положим

$$G = \text{'}pa\text{'}, \quad H = \text{'}gu\text{'} \quad K = \text{'}no\text{'},$$
$$\bar{G} = \text{'}es\text{'}, \quad \bar{H} = \text{'}li\text{'}, \quad \bar{K} = \text{'}r\text{'}.$$

Тогда, если $X = $ '*papagueno*', то

$$Y = \text{'}espalier\text{'}.$$

Заметим, что мы заменили на '*es*' только первое вхождение '*pa*', соответствующее G, но не тронули второе вхождение '*pa*', соответствующее P.

Шестерка $\langle G, H, K; \bar{G}, \bar{H}, \bar{K}\rangle$ представляет собой *схему продукций*. Мы будем обозначать конкретную *продукцию*, поставленную в соответствие этой шестерке посредством слов P и Q, следующим образом:

$$GPHQK \to \bar{G}P\bar{H}Q\bar{K}. \tag{1}$$

Будем говорить, что слово $Y = \bar{G}P\bar{H}Q\bar{K}$ есть *следствие* слова $X = GPHQK$ относительно продукции (1).

Продукция

$$\bar{G}P\bar{H}Q\bar{K} \to GPHQK \tag{2}$$

есть продукция, *обратная* к продукции (1).

Выражение $GPHQK \to \bar{G}P\bar{H}Q\bar{K}$ нередко используют для обозначения схемы продукций и говорят о нем просто как о «продукции»; в таком случае P и Q должны рассматриваться как переменные, принимающие значения из некоторых множеств слов.

3.1.2. Определения

Чтобы определить комбинаторную систему, необходимо задать:

1) Конечный алфавит $\mathfrak{A}$, называемый *алфавитом системы*. Для записи продукций может понадобиться также *вспомогательный алфавит* $\mathfrak{B}$.

2) Выделенное непустое слово, называемое *аксиомой* системы.

3) Конечное число схем продукций.

Таким образом, комбинаторная система — это формальная система, в которой

. все слова над $\mathfrak{A}$ (или, быть может, над $\{\mathfrak{A} \cup \mathfrak{B}\}$) являются формулами;

.. множество элементарных теорем содержит ровно одно слово — аксиому;

... правила вывода задаются продукциями.

Примеры мы приведем после того, как определим некоторые частные типы комбинаторных систем.

3.1.3. Продукции специальных типов

Пусть имеется шестерка $\langle E, H, E; E, \overline{H}, E\rangle$ (частный случай шестерки $\langle G, H, K; \overline{G}, \overline{H}, \overline{K}\rangle$ на стр. 49), где $|H| \neq 0$. Продукции вида

$$EPHQE \rightarrow EP\overline{H}QE$$

называются *полутуэвскими*[1]); продукции, обратные к полутуэвским, также являются полутуэвскими.

Пустое слово можно и не писать. Тогда мы имеем

$$PHQ \rightarrow P\overline{H}Q.$$

Если это не ведет к недоразумениям, мы можем сократить и эту запись:

$$H \rightarrow \overline{H}.$$

Теперь возьмем другой частный случай нашей исходной шестерки, а именно:

$$\langle G, E, E; E, E, \overline{K}\rangle,$$

которую можно записать, поменяв обозначения, как

$$\langle G, E, E; E, E, \overline{G}\rangle.$$

Это даст нам продукции вида

$$GPEQE \rightarrow EPEQ\overline{G},$$

или

$$GPQ \rightarrow PQ\overline{G},$$

и в конце концов (поскольку P и Q можно объединить)

$$GP \rightarrow P\overline{G}.$$

Подобные продукции называются *нормальными*, а обратные к ним продукции

$$P\overline{G} \rightarrow GP$$

называются *антинормальными*; при этом предполагается, что

$$|G||\overline{G}| \neq 0.$$

Пример. Пусть $G =$ '*то*', $\overline{G} =$ '*рос*'; тогда нормальная схема $GP \rightarrow P\overline{G}$ при $P =$ '*мат*' дает

'*томат*' $\rightarrow$ '*матрос*'.

Антинормальная схема $P\overline{G} \rightarrow GP$ дает при тех же условиях

'*матрос*' $\rightarrow$ '*томат*'.

---

[1]) Термин «туэвские» используется применительно к правилам без направления, поэтому здесь понадобилась приставка «полу».

3.1.4. Частные случаи комбинаторных систем

Полутуэвская система — это система, содержащая только полутуэвские продукции.

Туэвская система — это полутуэвская система, в которой для каждой продукции имеется обратная.

Нормальная система — это система, содержащая только нормальные продукции.

Система Поста — это система, содержащая только нормальные и антинормальные продукции, в которой для каждой продукции имеется обратная.

3.1.5. Примеры полутуэвских систем

Пример 1. Алфавит системы $\mathfrak{A} = \{a, b\}$, вспомогательный алфавит $\mathfrak{B} = \{s\}$, аксиома — слово 's'. Система содержит две полутуэвские продукции:

$$s \to ab, \tag{1}$$

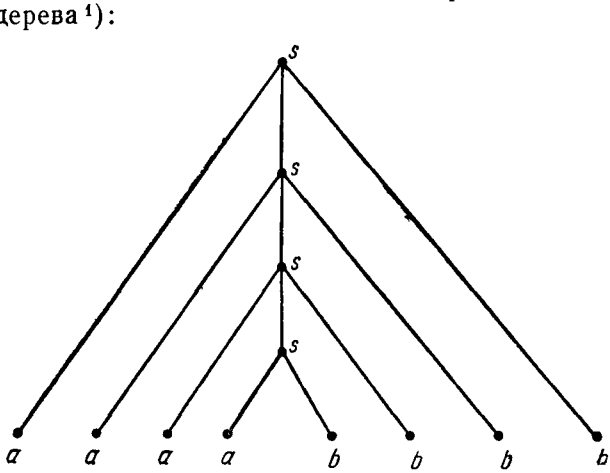
$$s \to asb. \tag{2}$$

Один из возможных выводов:

$$s \xrightarrow{(2)} asb \xrightarrow{(2)} aasbb \xrightarrow{(2)} aaasbbb \xrightarrow{(1)} aaaabbbb.$$

Последнее слово (a^4b^4) записано целиком в алфавите системы; вывод не может быть продолжен.

Данный вывод слова *aaaabbbb* можно представить в виде следующего дерева[1]):



Возможность такого представления вывода слова в виде дерева связана с тем, что в левых частях продукций рассматриваемой

[1]) Это представление — принципиально иное, чем описанное на стр. 54—55 (ср. примечание на стр. 55). — *Прим. ред*

мой системы возможны только однобуквенные слова. Если, например, ввести продукцию

$$aabb \to ba, \tag{3}$$

то вывод уже нельзя будет изобразить этим способом.

Пример 2. В этом примере мы будем использовать некоторые лингвистические термины: вместо «алфавит» мы будем говорить «словарь», вместо «слово» — «фраза» и т. п.

Словарь системы

$\mathfrak{W} = \{l', un, enfant, abricot, cueille, mange\}$.

Вспомогательный словарь содержит следующие символы:

Ph — фраза,
Gn — именная группа,
Gv — глагольная группа,
Art — артикль,
S — существительное,
V — глагол.

Аксиома системы — символ *Ph*[1]).

Система содержит следующие полутуэвские продукции:

$Ph \to Gn\ Gv$		(1)
$Gn \to Art\ S$		(2)
$Gv \to V\ Gn$		(3)
$Art \to l'$		(4)
$Art \to un$		(4′)
$S \to enfant$	'ребенок'	(5)
$S \to abricot$	'абрикос'	(5′)
$V \to cueille$	'срывает'	(6)
$V \to mange$	'ест'	(6′)

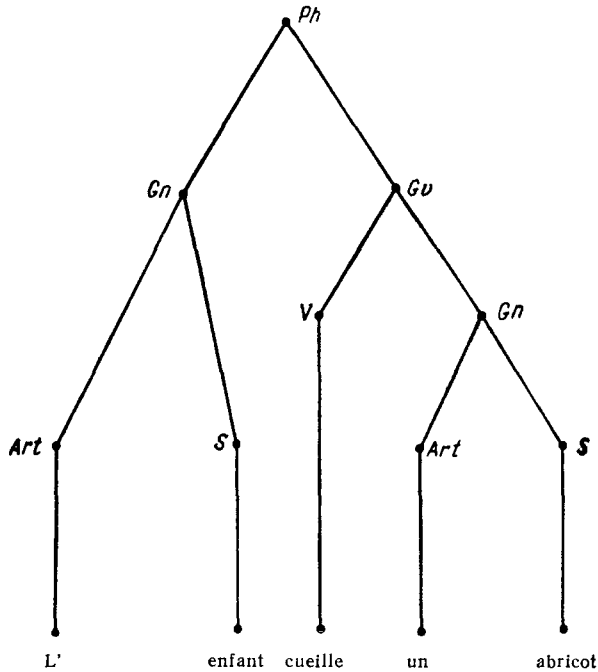
Первые три правила соответствуют в общих чертах правилам французского синтаксиса: «чтобы получить фразу, напишите именную группу, а за ней — глагольную группу», «чтобы получить именную группу, напишите артикль, а за ним — существительное» и т. д.

Остальные правила позволяют переходить от некоторых классов слов к конкретным словам этих классов.

---

[1]) Подчеркнем, что выражение *Ph*, равно как и выражения *Gn*, *Gv*, *Art*, — единый символ, т. е. одна буква вспомогательного алфавита, а не цепочка букв, ср. идентификаторы в Алголе. — *Прим. перев.*

Ниже изображена в виде дерева процедура построения фразы L'enfant cueille un abricot «Ребенок срывает абрикос» (возможность представить эту процедуру посредством дерева объясняется теми же соображениями, что и в предыдущем примере):



Аналогичным образом мы могли бы построить фразы Un enfant mange l'abricot «Ребенок ест абрикос», L'abricot cueille l'enfant «Абрикос срывает ребенка» и т. п. Первая из этих фраз — правильная и вполне банальная французская фраза, вторая же могла бы фигурировать в сказке вроде «Алисы в стране чудес».

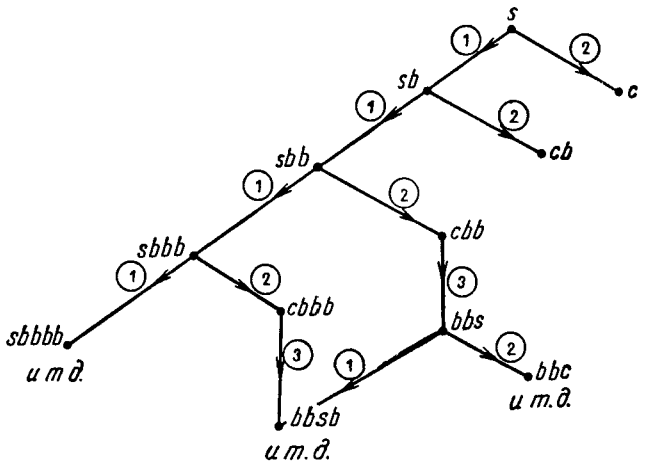
Пример 3. Алфавит системы — $\{s, b, s\}$; аксиома — 's'; продукции

$$s \to sb, \qquad (1)$$
$$s \to c, \qquad (2)$$
$$cbb \to bbs \qquad (3)$$

являются полутуэвскими.

Из-за продукции (3) вывод в данной системе уже нельзя представить в виде дерева, как в предыдущих примерах. Однако он может быть изображен посредством ориентированного графа, устроенного следующим образом: на каждой дуге графа поме-

щается номер примененной продукции; ребра с (1) направлены налево, ребра с (2) — направо, а ребра с (3) — вертикально вниз.



Заметим, что к слову *bbsb* ведут из *s* два пути [1]).

3.1.6. Доказательство теорем

Пусть имеется комбинаторная система Г и конечная последовательность Σ слов

$$X_1, X_2, \ldots, X_m,$$

где X_1 — выделенное слово, т. е. аксиома системы Г, а каждое слово X_j, $1 < j \leqslant m$, есть следствие из слова X_{j-1} относительно некоторой продукции из Г.

Мы будем называть последовательность Σ *доказательством* в Г, а последнее слово последовательности Σ — *теоремой* в Г.

Переход от слова X_{j-1} к слову X_j называется *шагом* доказательства. (М. Дэвис называет «шагом» каждое слово X_j, что представляется менее естественным.)

Вместо «доказательство теоремы» мы будем говорить также «вывод слова» (или «вывод фразы», если базовое множество названо «словарем») [2]).

---

[1]) Данный граф в отличие от деревьев предыдущих примеров представляет не один вывод, а все множество выводов в данной системе; представлениями отдельных выводов являются пути в графе. В частности, наличие двух путей из *s* в *bbsb* означает, что слово *bbsb* имеет два различных вывода Заметим еще, что так можно представить множество выводов *любой* комбинаторной системы. — *Прим. ред.*

[2]) Введенное здесь понятие доказательства может рассматриваться как частный случай (для комбинаторных систем) общего понятия вывода, определенного на стр. 47. Действительно, описанное там дерево в случае комбинаторной системы будет деревом без ветвления (поскольку каждое правило имеет теперь лишь одну посылку), т. е. линейной последовательностью. — *Прим. ред.*

Иногда приходится доказывать некоторое утверждение обо всех словах данной системы или обо всех словах определенного класса. В тех случаях, когда возможно неоднозначное понимание, подобное утверждение будет в отличие от теорем системы называться *метатеоремой*; язык, на котором формулируется метатеорема, является по отношению к рассматриваемой системе *метаязыком.*

Пример 1. Вернемся к примеру 1 из п. 3.1.5. Алфавит системы — $\{a, b\}$, аксиома — 's', продукции — $s \to ab$, $s \to asb$. Мы доказали теорему *aaaabbbb* и привели дерево доказательства.

Утверждение «Всякое выводимое из аксиомы слово системы, не содержащее буквы s, представимо как конкатенация слова, состоящего из вхождений буквы a, со словом той же длины, состоящим из вхождений буквы b» есть метатеорема.

Пример 2. В примере 3 из п. 3.1.5 слово '*bbsb*' является теоремой, имеющей, как это видно из графа, иллюстрирующего пример, два разных доказательства.

Утверждение «В данной системе существует бесконечно много теорем, каждая из которых имеет два доказательства» есть метатеорема.

3.1.7. Моногенные системы

Комбинаторная система Г называется *моногенной*, если к любой ее теореме может быть применено не более одной продукции. Иначе говоря, в моногенной системе Г невозможна теорема X, такая, что

. X представима двумя разными способами — как $GPHQK$ и как $G'P'H'Q'K'$, причем

.. в Г имеются две продукции с левыми частями $GPHQK$ и $G'P'H'Q'K'$ соответственно.

Пример 1. Система примера 1 из п. 3.1.5 не является моногенной, потому что к s можно применить две разные продукции, иначе говоря, из s можно получить два разных следствия, а именно ab и asb.

Пример 2. Система с алфавитом $\{a, b, c\}$, вспомогательным алфавитом $\{s, t, u\}$, аксиомой 's' и продукциями

$$s \to atb,$$
$$atb \to aaub,$$
$$u \to c$$

является моногенной и позволяет получить лишь конечное число теорем.

Пример 3. Система с алфавитом $\{a, b, s\}$, аксиомой 's' и продукцией $s \to asb$ является моногенной и позволяет получить бесконечное число теорем.

3.1.8. Неоднозначность

Встав на наиболее общую точку зрения, мы будем считать два доказательства одной и той же теоремы различными, если они различаются либо входящими в них словами, либо порядком этих слов, либо и тем, и другим.

Теорема (выводимое из аксиомы слово или фраза) является *неоднозначной*, если для нее существует два различных доказательства (вывода).

Для приложений можно вводить различные — более сильные или более слабые — понятия неоднозначности.

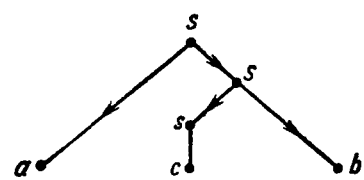
Пример. С ситуацией неоднозначности слова мы встречались в примере 3 п. 3.1.5: к слову *bbsb* можно было прийти двумя путями. Вот еще один пример.

Пусть имеется алфавит $\{a, b, c, s\}$, аксиома 's' и следующие продукции:

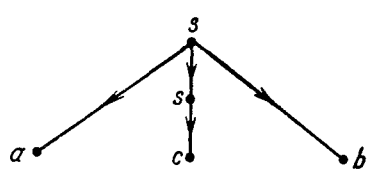
$$s \to as \qquad (1)$$
$$s \to sb \qquad (2)$$
$$s \to asb \qquad (3)$$
$$s \to c \qquad (4)$$

Теорема acb является неоднозначной. Действительно, она имеет два доказательства:

1) $s \xrightarrow{(1)} as \xrightarrow{(2)} asb \xrightarrow{(4)} acb,$



2) $s \xrightarrow{(3)} asb \xrightarrow{(4)} acb.$



§ 3.2. НОРМАЛЬНЫЕ СИСТЕМЫ

3.2.0. Нормальная система

Чтобы определить *нормальную систему*, необходимо задать

. алфавит $\mathfrak{A} = \{a_i \mid 1 \leqslant i \leqslant n\}$;

.. аксиому A;

... ν *нормальных продукций* (схем продукций)

$$\{G_j P \to P \overline{G}_j \mid 1 \leqslant j \leqslant \nu\},$$

где A, G_j и $\overline{G}_j$ — слова свободной полугруппы, порождаемой множеством $\mathfrak{A}$.

Пример. Пусть имеется нормальная система с алфавитом $\{a, b\}$, аксиомой 'a' и схемами продукций

$$aP \to Pbba,$$
$$bP \to Paba.$$

Пример доказательства: $a \to bba \to baaba.$

3.2.1. Каноническое соответствие между нормальной и полутуэвской системами

Для упрощения записи мы будем рассматривать систему с алфавитом из трех букв; легко, однако, видеть, что общность наших рассуждений нисколько от этого не страдает.

Итак, пусть имеется полутуэвская система (ST) [1]:

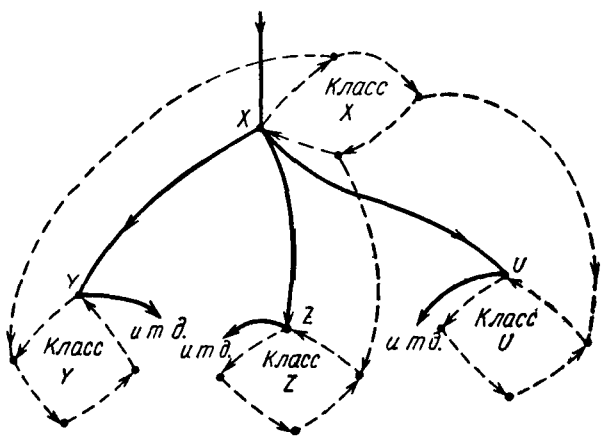
$$(ST) = \begin{cases} \text{алфавит:} & \mathfrak{A} = \{a, b, c\}, \\ \text{аксиома:} & A, \\ \text{продукции:} & PG_iQ \to P\bar{G}_iQ, \qquad 1 \leqslant i \leqslant m. \end{cases}$$

Требуется построить по системе (ST) такую нормальную систему (N), которая имела бы ту же аксиому A и теоремы которой группировались бы в классы в соответствии с некоторым отношением эквивалентности; при этом теоремы системы (N) должны быть связаны с теоремами системы (ST) следующим образом:

. всякая теорема системы (ST) является теоремой и в (N);

·· всякая теорема системы (N) принадлежит некоторому классу эквивалентности, в котором имеется и некоторая теорема системы (ST).

Указанное соответствие между системами (ST) и (N) иллюстрируется следующей схемой:



X, Y, Z, U — теоремы в (ST), → — полутуэвские продукции, ⇢ — нормальные продукции

[1]) ST — от semi-thueien. — *Прим. перев.*

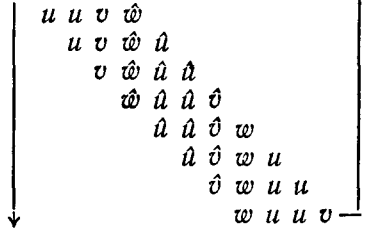
Чтобы построить (N), мы продублируем алфавит системы (ST), введя в него буквы со штрихами: a', b' и c'. Всякому слову M, записанному в алфавите $\mathfrak{A}$, отвечает слово M', получаемое из M добавлением штрихов ко всем вхождениям букв в M; условимся, что $E' = E$.

Теперь мы можем определить

$$(N) = \begin{cases} \text{алфавит: } \mathfrak{B} = \{a, b, c, a', b', c'\}, \\ \text{аксиома: } A, \\ \text{продукции трех типов:} \\ 1) \qquad aP \to Pa' \text{ и аналогичные,} \\ 2) \qquad a'P \to Pa \text{ и аналогичные,} \\ 3) \qquad G_iP \to P\bar{G}_i', \ 1 \leqslant i \leqslant m. \end{cases}$$

Сопряженные слова. Заметим, что продукции типов (1) и (2) построенной нами системы (N) позволяют перебрасывать некоторую букву из начала слова в его конец, меняя при этом ее «штриховость» (если она имела штрих, он утрачивается; если штриха не было, он появляется).

Пусть u, v, w, ... — переменные, представляющие буквы алфавита $\mathfrak{B}$, а $\hat{u}$, $\hat{v}$, $\hat{w}$, ... — переменные, представляющие те же буквы, но с обратной «штриховостью». Возьмем, например, слово $wuuv$ и будем поочередно применять к нему ту из продукций типа (1) или (2), которую удается применить. В результате получится циклический вывод:



Таким образом, можно определить класс эквивалентности, состоящий из *сопряженных слов*. В частности, среди всех слов, сопряженных со словом M, есть и M'. Это можно усмотреть из вывода на приведенной выше схеме — для этого достаточно интерпретировать наличие (отсутствие) крышечки над буквой как наличие (отсутствие) штриха.

Все слова, сопряженные со словами из $\mathfrak{A}^*$, и только эти слова, имеют форму PQ' или $P'Q$, где P, $Q \in \mathfrak{A}^*$.

Такие слова мы назовем *регулярными*.

Прямое утверждение. Докажем, что всякая теорема полутуэвской системы (ST) является теоремой нормальной системы (N).

Лемма 1. *Всякое слово, сопряженное с теоремой системы* (N), *есть теорема системы* (N).

В самом деле, слова, сопряженные с теоремой, выводятся из нее посредством применения продукций системы (N).

Заметим, что доказываемое нами утверждение явным образом верно для аксиомы A, общей обеим системам. Поскольку все теоремы получаются из A, достаточно доказать, что свойство «быть теоремой системы (N)» сохраняется при применении продукций системы (ST).

Предположим, что теорема X системы (ST) является теоремой в (N). Для того чтобы X имела следствие в (ST), необходимо, чтобы X могла быть представлена в виде $X = PG_iQ$: тогда из X непосредственно следует $\overline{X} = P\overline{G}_iQ$.

Однако в силу леммы 1 слово G_iQP', сопряженное с X, является теоремой (N). Применяя к G_iQP' подходящую продукцию типа (3), мы можем получить слово $QP'\overline{G}_i'$, также являющееся теоремой (N). Это последнее слово сопряжено с $P\overline{G}_iQ$, которое поэтому (в силу леммы 1) есть теорема системы (N), что и требовалось доказать.

Обратное утверждение. *Всякая теорема системы* (N) *является регулярным словом, сопряженным с некоторой теоремой системы* (ST).

Будем действовать аналогично предыдущему. Аксиома A системы (N) является регулярным словом, сопряженным с некоторой теоремой системы (ST), а именно с самим собой. Поэтому достаточно доказать, что оба свойства (регулярность и сопряженность с некоторой теоремой системы (ST)) сохраняются при применении продукций системы (N).

Сохранение указанных свойств очевидно для продукций типов (1) и (2); рассмотрим продукции типа (3).

Предположим, что теорема G_iP регулярна и сопряжена с некоторой теоремой системы (ST). Так как $G_i \neq E$, из регулярности G_iP следует, что она может быть записана в виде $G_iP_1P_2'$, где P_1 и P_2 — слова в алфавите системы (ST). Единственное сопряженное с G_iP слово в алфавите системы (ST) — это $P_2G_iP_1$; из нашего предположения вытекает, что $P_2G_iP_1$ — теорема (ST). Отсюда следует, что $P_2\overline{G}_iP_1$ — также теорема (ST); однако $P_2\overline{G}_iP_1$ сопряжена со словом $P_1P_2'\overline{G}_i'$, которое является регулярным.

Итак, мы показали, что слово $P\overline{G}_i'$, полученное из G_iP посредством продукции типа (3) системы (N), сохраняет регулярность

и сопряженность с некоторой теоремой (ST). Доказательство закончено.

Резюме. Пусть имеется полутуэвская система с алфавитом $\{a, b, c, \ldots\}$. Мы умеем поставить ей в соответствие нормальную систему с алфавитом $\{a, a', b, b', c, c', \ldots\}$, такую, что множество ее теорем, не содержащих штрихованных букв, совпадает со множеством теорем исходной полутуэвской системы.

§ 3.3. УПРАЖНЕНИЯ

1. Дана система с алфавитом $\mathfrak{A} = \{a, b, c\}$, аксиомой '$a$' и продукциями $a \to b$, $b \to c$ и $c \to a$.

Является ли эта система моногенной?

Является ли теорема 'c' неоднозначной?

Исключается ли неоднозначность моногенностью?

2. В примере п. 3.1.8 слово 'acb' неоднозначно из-за возможных *ветвлений* при построении вывода из 's'. Рассмотрим систему с алфавитом $\mathfrak{A} = \{a, b, s\}$, аксиомой '$s$' и продукциями $s \to ab$ и $s \to asb$. Является ли она моногенной? Получаются ли в ней неоднозначные теоремы?

Является ли наличие ветвления достаточным условием для неоднозначности?

3. Дана система, имеющая продукции вида $a \to a$ и $b \to E$. Основываясь на этом конкретном примере, подвергнуть критике предложенное нами весьма общее определение неоднозначности. Подвергнуть его критике также с другой точки зрения, используя пример системы

$$\Gamma = \begin{cases} \mathfrak{A} = \{A,\ B,\ C,\ b,\ c\} \\ \text{аксиома: } A \\ \text{продукции: } A \to BC,\ B \to b,\ C \to c. \end{cases}$$

4. Построить способом, предложенным в п. 3.2.1, нормальную систему для полутуэвской системы

$$(ST) = \begin{cases} \mathfrak{A} = \{a,\ b,\ c\} \\ \text{аксиома: } 'c' \\ \text{продукция: } PcQ \to PacbQ. \end{cases}$$

§ 3.4. НЕЗАВИСИМОСТЬ ОТ АЛФАВИТА

Известно, что буквы всякого алфавита могут быть закодированы с помощью букв другого алфавита меньшей мощности. Алфавит Морзе, позволяющий кодировать любые алфавиты, содер-

жит всего три элементарных символа: точка, тире, пробел. При одних способах кодирования пробел представляется специальным символом, при других он получается «сам собой».

В электронных вычислительных машинах используется двоичный код; с помощью слов, состоящих из n вхождений двоичных символов, можно закодировать 2^n букв. Специальный символ для пробела не нужен благодаря делению машинной памяти на ячейки.

3.4.1. Двоичное кодирование

Ясно, что для любой комбинаторной системы (S) можно построить эквивалентную систему (S^+) с алфавитом $\{0, 1\}$. Вот один из возможных способов построения такой системы.

Пусть $\mathfrak{A} = \{a_1, \ldots, a_n\}$ есть алфавит системы (S). Положим $a_i^+ = \underbrace{\text{'1000} \ldots \text{01'}}_{i+1}$; иначе, $a_i^+ = \text{'}10^{i+1}1\text{'}$.

Слову $M = a_{i_1} \ldots a_{i_p}$ мы поставим в соответствие слово $M^+ = a_{i_1}^+ \ldots a_{i_p}^+$, а пустому слову E — пустое слово E^+. Таким образом, полугруппа $\mathfrak{A}^*$ отображается на некоторое подмножество M^+ множества $\{0, 1\}^*$; это отображение взаимно однозначно. Более точно, если дано слово в алфавите $\{0, 1\}$, то мы умеем

. узнать, принадлежит ли оно M^+;

.. в случае, если оно принадлежит M^+, найти единственное слово в M, являющееся его прообразом.

Будем говорить, что некоторое слово в алфавите $\{0, 1\}$ является *правильно построенным,* если у него есть прообраз в $\mathfrak{A}^*$.

3.4.2. Построение системы S^+

Если (S) имеет аксиому A и продукции

$$G_i P H_i Q K_i \to \bar{G}_i P \bar{H}_i Q \bar{K}_i \mid i = 1, \ldots, m,$$

то в качестве аксиомы и продукций для (S^+) мы возьмем соответственно A^+ и

$$G_i^+ P H_i^+ Q K_i^+ \to \bar{G}_i^+ P \bar{H}_i^+ Q \bar{K}_i^+ \mid i = 1, \ldots, m.$$

Предложение 1. *Если M — теорема системы (S), то M^+ — теорема системы (S^+).*

В самом деле, если $M_1, \ldots, M_q, M$ есть доказательство M в (S), то $M_1^+, \ldots, M_q^+, M^+$ есть доказательство M^+ в (S^+), что и требовалось доказать.

Предложение 2. *Всякая теорема системы (S^+) является правильно построенным словом.*

Аксиома системы (S^+) обладает этим свойством, и продукции системы (S^+) сохраняют его, что и требовалось доказать.

Предложение 3. *Всякая теорема системы* (S^+) *имеет своим прообразом некоторую теорему системы* (S).

Возьмем произвольную теорему системы (S^+). Поскольку она является правильно построенным словом, она имеет вид M^+ при $M \in \mathfrak{A}^*$. Точно так же обстоит дело со всеми теоремами, фигурирующими в ее доказательстве. При этом, если некоторая теорема B системы (S^+) получается из теоремы A системы (S^+) с помощью некоторой продукции этой системы, то прообраз B получается из прообраза A с помощью соответствующей продукции системы (S). Поэтому доказательство теоремы M^+ в (S^+) дает доказательство теоремы M в (S), что и требовалось доказать.

3.4.3. Разрешимость в комбинаторных системах

Используя гёделевскую нумерацию[1]) и вводя необходимые предикаты, можно доказать, что множество теорем любой комбинаторной системы рекурсивно перечислимо.

Это свойство еще не позволяет для конкретной системы сказать, является ли множество ее теорем рекурсивным; если же это так, система называется *разрешимой*.

Разрешимость системы не зависит от алфавита в том смысле, что любой системе (S) можно поставить в соответствие систему (S^+) с алфавитом всего из двух символов, для которой проблема разрешимости имеет то же решение, что и для (S).

3.4.4. Упражнения

1. Что можно сказать о (S^+) в следующих случаях: система (S) является полутуэвской, туэвской, нормальной, системой Поста?

2. Найти более экономный способ кодирования, нежели

$$a_i^+ = 10^{i+1}1,$$

показав, что не обязательно иметь два маркера.

3. Закодируем алфавит $\{a, b, c, d\}$ следующим образом: $a \to 0$, $b \to 10$, $c \to 110$, $d \to 111$. Необходим ли символ-разделитель? Ответить на этот же вопрос при кодировках $a \to 00$, $b \to 01$, $c \to 10$, $d \to 11$ и $a \to 0$, $b \to 1$, $c \to 10$, $d \to 11$.

---

[1]) О гёделевской нумерации см. ниже § 5.3. — *Прим. перев.*

Глава IV

АЛГОРИТМЫ. МАШИНЫ ТЬЮРИНГА

§ 4.1. АЛГОРИТМЫ

4.1.1. Понятие алгоритма [1])

В начальной школе проходят, как известно, четыре действия арифметики. Научившись, например, умножению, школьник умеет перемножать *любые* числа: по существу он усвоил алгоритм умножения.

На интуитивном уровне, алгоритм — это система правил, позволяющих чисто механически выполнить любую конкретную операцию некоторого определенного типа. Можно охарактеризовать алгоритм еще и как такую механическую процедуру, которая может применяться к символам некоторого класса (*входным* символам) и, возможно, выдает для данного входного символа определенный *выходной* символ.

Пример 1. *Дифференцирование многочлена.*

(1) Взять очередной член многочлена; если такового уже не осталось, написать «конец» и остановиться.

(2) Взять показатель степени переменной данного члена и

(2.1) умножить его на коэффициент; сделать это произведение коэффициентом при новом члене;

(2.2) уменьшить его на 1 и сделать показателем степени при новом члене.

(3) Вернуться к (1).

Пример 2. *Поиск слова в двуязычном (переводном) словаре.*

(1) Взять первую букву искомого слова и найти тот раздел словаря, первое слово которого начинается этой буквой.

(2) Взять первые три буквы искомого слова и найти в найденном на шаге (1) разделе первую из страниц, на которых имеются слова, начинающиеся этими буквами.

(3) Начиная с найденной страницы, просматривать подряд все слова, сравнивая их с искомым словом до тех пор, пока либо будет получено совпадение («искомое слово найдено»), либо мы дойдем до слова, не начинающегося на три буквы, выделенные на шаге (2) («искомое слово не найдено»).

(4) Если искомое слово найдено, взять все его эквиваленты; конец.

---

[1]) Более подробно с этим понятием можно познакомиться по статье «Алгоритм» в «Философской энциклопедии», т. 1, М., 1960, или по книге Б. А. Трахтенброта «Алгоритмы и машинное решение задач», М., 1960. — *Прим. ред.*

(5) Если искомое слово не найдено, взять более полный словарь и вернуться к (1) [1]).

Процедуры, описанные в обоих примерах, являются механическими (в интуитивном смысле этого слова). Однако, если бы мы захотели реализовать эти процедуры с помощью машины, лишенной сообразительности, знаний о мире и т. д., нам пришлось бы заняться их уточнением: многие инструкции в данном описании не сформулированы в явном виде.

Более детальная характеристика понятия «алгоритм» должна включать следующие аспекты (по Хартли Роджерсу):

а) Алгоритм есть совокупность инструкций, имеющих *конечную длину* (эта длина измеряется, например, числом символов, необходимых для записи правил).

б) Имеется механизм (человек, механическое, оптическое и т. п. устройство, электронная машина), воспринимающий и выполняющий инструкции.

в) Имеются средства (человеческая память, карандаш и бумага, зубчатые колеса, магнитные запоминающие устройства и т. п.), позволяющие фиксировать и хранить сведения о любых этапах работы по выполнению инструкций, а также выдавать эти сведения по мере необходимости.

г) Существенно, что *все выполняемые процедуры являются дискретными* (хотя обрабатываемые ими объекты могут быть по своему характеру непрерывными).

д) *Последовательность элементарных операций,* из которых складываются инструкции, *жестко определена* (как обычно говорят, *детерминирована*): на каждом шаге процесса переход к следующему шагу возможен не более чем одним способом.

Напрашивается аналогия с парой ⟨вычислительная машина, программа⟩, а именно:

а) соответствует программе;

б) соответствует вычислительной машине, работающей по этой программе;

в) соответствует машинной памяти;

г) соответствует дискретной («цифровой») природе цифровых вычислительных машин (в отличие от аналоговых);

д) соответствует механичности и жесткой детерминированности порядка выполнения команд программы.

---

[1]) Описанная здесь процедура годится, разумеется, лишь для простейшего случая, когда искомое слово имеет словарную форму (например, для русского языка, существительные — в им. пад. ед. числа, глаголы — в инфинитиве и т. д.). Более общая ситуация (для языков, где имеется словоизменение) предполагает предварительное приведение текстового слова к словарному виду. — *Прим. перев.*

4.1.2. Алгоритмы и формальные грамматики

Понятие алгоритма используется в формальной лингвистике, в частности, следующим образом.

Выше (стр. 23) мы определили (формальный) язык L над алфавитом $\mathfrak{A}$ (над словарем $\mathfrak{B}$): L есть множество слов (фраз), принадлежащих свободной полугруппе $\mathfrak{A}^*$ (соответственно $\mathfrak{B}^*$) с системой образующих $\mathfrak{A}$ (соответственно $\mathfrak{B}$). Язык L представляет собой множество допустимых, т. е. (синтаксически) правильных слов (фраз). Поэтому $\mathfrak{A}^* \setminus L$ есть множество недопустимых, т. е. неправильных слов (фраз).

Формальная грамматика должна дать правила, которые обеспечивают либо получение правильно построенных слов (правильных фраз), либо распознавание того, является ли данная цепочка букв (слов) правильно построенным словом (правильной фразой). Другими словами, цель грамматики состоит в том, чтобы как-то задать (охарактеризовать) слова (фразы), принадлежащие данному языку. Таким образом, формальная грамматика предстает как алгоритм, а именно, как совокупность правил, позволяющих задать некоторый язык[1]).

Если предположить, что у нас имеется алгоритм (в смысле, указанном в предыдущем пункте), такой, что его входными символами являются имена целых чисел $1, 2, \ldots, n$ и каждому входному символу отвечает на выходе некоторое слово (фраза) языка, определяемого данным алгоритмом, то мы можем сказать, что этот алгоритм — или грамматика — *перечисляет* слова (фразы) этого языка.

Пример 1. Алфавит $\mathfrak{A} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$; слово принадлежит языку L тогда и только тогда, когда оно является десятичным представлением простого числа.

Алгоритм, задающий язык L, легко себе представить; он может быть основан, например, на решете Эратосфена.

Пример 2. Алфавит $\mathfrak{A} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$; язык L определяется следующим образом.

Рассмотрим (бесконечное) десятичное разложение числа π. Цепочка цифр является словом языка L тогда и только тогда, когда она встречается в этом разложении между двумя последователь-

[1]) Утверждение, что грамматика — это алгоритм, содержит неточность. В математической логике различаются понятия алгоритма и исчисления, в наиболее общем виде можно, следуя А. А. Маркову, охарактеризовать алгоритм как *предписание*, а исчисление — как *разрешение* производить некоторые действия. В частности, (детерминированная) машина Тьюринга — это один из способов реализации алгоритма, а комбинаторные системы — это исчисления. Грамматики, как они здесь (и обычно) определяются, представляют собой исчисления, поскольку в них нет ограничений на порядок применения правил. Чтобы превратить грамматику в алгоритм, необходимо присоединить к ней какой-либо механизм, устанавливающий жесткий порядок применения правил. Ср. Гладкий — Мельчук [1969], стр. 45—46. — *Прим. ред.*

ными вхождениями символа 0. Алгоритм, задающий этот язык, основывается на вычислении числа π.

Естественным образом возникает следующая *алгоритмическая проблема*:

По данной цепочке вхождений букв (слов) узнать, является ли она правильно построенным словом (правильной фразой) данного языка.

Пример. Всякий язык программирования состоит из исходных символов и правил образования выражений; правила приводятся в инструкции по использованию языка. (Если это хорошая инструкция, она представляет собой грамматику данного языка.) Правильная программа является правильной фразой языка, и обратно.

Компилирование программы, написанной на некотором языке программирования, для определенной конкретной машины — это операция, которую достаточно трудно определить формально. Грубо говоря, мы можем рассматривать компилирование программ как «перевод» с одного языка на другой: правильным цепочкам входного языка должны соответствовать правильные цепочки выходного языка. Процесс компилирования включает *проверку* синтаксической правильности входной программы. Эта проверка осуществляется специальным алгоритмом, имеющим два особых выходных символа, содержательный смысл которых — «правильно» и «неправильно».

Легко убедиться, что наличие алгоритма перечисления цепочек еще не гарантирует существования алгоритма распознавания их правильности. В примере 1 этого пункта алгоритм, позволяющий узнать, является ли данное число простым, существует. Однако в примере 2 дело обстоит иначе: по крайней мере, насколько известно авторам, пока что никем не предложен алгоритм, позволяющий узнать, является ли данная цепочка словом из L. Быть может, когда-нибудь удастся доказать, что такой алгоритм вообще нельзя построить.

До сих пор мы имели дело с чисто интуитивным понятием алгоритма. Его, однако, можно формализовать, что и было в свое время сделано с помощью ряда конструкций, которые в конце концов оказались эквивалентными. В частности, для формализации понятия «алгоритм» привлекались комбинаторные системы, о которых шла речь выше, и так называемые машины Тьюринга, которые будут рассматриваться в этой главе.

4.1.3. Вычислимость

С помощью понятия машины Тьюринга формализуется интуитивное понятие вычислимости (а тем самым — интуитивное понятие алгоритма). Мы имеем в виду прежде всего психологическое пред-

ставление о вычислимости: вычислителю (обычно человеку) предлагаются некоторые исходные данные и сообщаются правила их обработки; он должен получить тот или иной результат, зависящий от этих данных. Нередко вычислитель-человек может заранее сказать, вычислим ли требуемый результат: ведь, как правило, он обладает определенным практическим опытом вычислительной работы. Так, известно, что

- число π не вычислимо, поскольку его десятичное представление бесконечно [1]);
- приближенное значение числа π с недостатком вычислимо с точностью до 10^{-n} при любом фиксированном n.

Однако в ряде случаев неизвестно, является ли требуемый результат вычислимым.

Заметим, что, говоря о вычислимости, мы отвлекаемся от естественных (физических) ограничений, характерных как для человека, так и для машины: для некоторых вычислений может понадобиться больше ячеек памяти, чем имеется элементарных частиц во всей нашей Галактике, и время, измеряемое миллиардами тысячелетий; тем не менее, невзирая на материальную неосуществимость подобных вычислений, можно видеть, что они представляют эффективный способ получить некоторый результат за *конечное* число шагов. Были предприняты некоторые математические исследования с целью выработать более реалистическое понятие вычислимости; однако, как бы то ни было, математические теории, имеющие дело с вычислимостью, не занимаются *практическими* вычислениями, такими, например, с какими имеет дело численный анализ.

Таким образом, представление о вычислимости, формализуемое с помощью точного понятия машины Тьюринга, является очень широким; мы увидим тем не менее, что даже в рамках этого широкого понятия некоторые объекты, которые интуитивно кажутся «вычислимыми», на самом деле таковыми не являются.

§ 4.2. МАШИНЫ ТЬЮРИНГА

Итак, мы хотим формализовать понятие вычислимости. Один из способов сделать это состоит в рассмотрении некоторой идеализированной вычислительной машины. При этом, поскольку мы хотим иметь возможность высказывать о ней как можно больше математических утверждений, она должна быть как можно более простой. Поэтому наша идеализированная машина не будет резко отличаться по своему строению от реальных электронных вычислительных машин (ЭВМ).

---

[1]) Обычно в теории алгоритмов термин «вычислимое действительное число» используется иначе — он обозначает число, для которого существует алгоритм нахождения его последовательных рациональных приближений с любой степенью точности, которая оценивается также эффективно. В этом смысле число π вычислимо. — *Прим. ред.*

Интересующий нас класс абстрактных машин был введен в рассмотрение еще в 1936 году (т. е. до создания первых электронных вычислительных машин) английским математиком и логиком А. Тьюрингом. И здесь, как во многих других областях, научная фантазия опередила действительность.

Впрочем, если говорить лишь о ее абстрактной схеме строения, оказывается, что машина Тьюринга не так уж далека от настоящих ЭВМ.

Верно, что у машины Тьюринга предполагается потенциально бесконечная память; однако если ЭВМ обладает внешней памятью на магнитной ленте или на дисках, причем мы располагаем достаточным количеством бобин ленты или запасом дисков, то ситуация оказывается практически такой же.

Верно, что машина Тьюринга читает за один ход лишь один символ; однако этот символ может иметь сложное значение.

Наконец, как и арифметическое устройство ЭВМ, машина Тьюринга имеет конечное число внутренних состояний. Ее поведение в каждый данный момент зависит от состояния, в котором она находится, и от того символа, который она получает в качестве внешней информации. Если отвлечься от того, что процесс ввода информации идеализирован и сведен к чтению *одного* атомарного символа на каждом шаге, то поведение машины Тьюринга не отличается от функционирования реальной ЭВМ.

4.2.1. Содержательное описание

Машина Тьюринга состоит из следующих частей:

. *Управляющее устройство*, способное принимать некоторое число *состояний*, которые соответствуют различным комбинациям состояний внутренних элементов электронной машины (ячейки памяти, электронные схемы и т. п.).

.. *Лента*, на которой заранее помещены подлежащие обработке данные и на которую заносятся в ходе работы машины промежуточные результаты вычислений.

Все записи на ленте производятся с помощью букв некоторого алфавита. Лента разделена на ячейки, в каждой из которых может находиться не более одного символа. Число ячеек предполагается неограниченным в обе стороны.

... *Головка*, выполняющая чтение и запись символов; она обеспечивает обмен информацией между лентой и управляющим устройством. Головка может работать в каждый момент только с одной ячейкой ленты. При этом она может

заменять прочитанный символ другим символом;

перемещать ленту влево или вправо на одну ячейку.

Машина Тьюринга работает следующим образом: в каждой ситуации, определенной внутренним состоянием управляющего устройства и символом, прочитанным на ленте, машина срабатывает,

переходя в другое состояние и передвигая ленту определенным образом. Это делается в соответствии с правилами перехода, которые указывают, как должна действовать машина в каждой заданной ситуации.

Уточним теперь (следуя Мартину Дэвису) представление о функционировании машин Тьюринга.

4.2.2. *Формальное описание*

Символы, которые машина читает и пишет на ленте, мы будем обозначать через $S_0, S_1, \ldots, S_n$; они образуют *алфавит* машины. S_0 интерпретируется как «пустой символ»; он играет особую роль: считается, что этот символ записан в каждой пустой ячейке, и, обратно, если в ячейке записан символ S_0, то она пуста. Мы будем писать вместо S_0 также B или #. Заменить S_i на S_0 означает стереть S_i.

Вместо S_1 мы часто будем писать «1» (единицу) или «|» (палочку).

Внутренние состояния (т. е. состояния управляющего устройства) будут обозначаться через

$$q_1, q_2, q_3, \ldots;$$

любая конкретная машина Тьюринга обладает конечным числом таких состояний.

Символы $П$ и $Л$ будут означать смещение ленты на одну ячейку вправо или влево соответственно [1]).

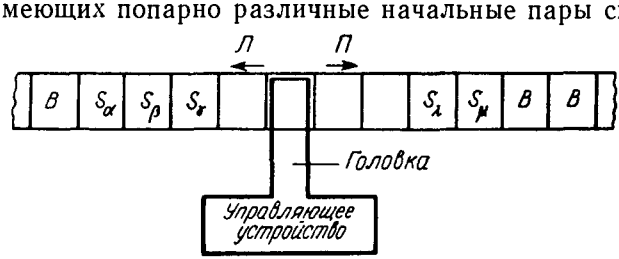
Дадим теперь несколько определений.

(1) *Выражением* называется конечная последовательность символов в алфавите $\{S_0, S_1, \ldots; q_1, q_2, \ldots; П, Л\}$.

(2) *Командой* называется выражение одного из трех следующих типов:

$$q_i\ S_j\ S_k\ q_l,$$
$$q_i\ S_j\ П\ q_l,$$
$$q_i\ S_j\ Л\ q_l.$$

(3) *Машиной Тьюринга* называется конечное множество команд, имеющих попарно различные начальные пары символов.



[1]) N3: В литературе символы $П$ и $Л$ часто употребляются в значениях, **противоположных** принятым здесь. Это связано с тем, что вместо перемещения ленты можно говорить о перемещении головки. — *Прим. ред.*

Работа машины Тьюринга. С помощью только что приведенных определений работа машины Тьюринга может быть теперь описана точно.

. Начальная пара некоторой команды задает *исходную ситуацию*: машина находится в состоянии q_i и читает символ S_j.

.. Поведение машины в данной *ситуации* q_iS_j полностью определяется заключительной парой символов той единственной команды, которая соответствует этой ситуации (т. е. начинается парой символов q_iS_j). При этом элементарные действия машины бывают одного из следующих трех типов:

Тип S_kq_l: машина заменяет S_j на S_k и переходит из состояния q_i в состояние q_l.

Тип $Пq_l$: машина сдвигает ленту на одну ячейку вправо и переходит из состояния q_i в состояние q_l.

Тип $Лq_l$: машина сдвигает ленту на одну ячейку влево и переходит из состояния q_i в состояние q_l.

Напомним, что на множество команд, задающих машину Тьюринга, наложено требование, чтобы начальные пары символов всех команд были различны. В силу этого требования в любой данной ситуации возможно не более чем одно очередное элементарное действие. Машину, для которой это требование выполнено, мы назовем *детерминированной*. Машину, для которой указанное требование не соблюдается, т. е. в некоторой ситуации она имеет право «выбирать» между несколькими разными элементарными действиями, мы будем называть *недетерминированной*.

Сформулируем еще одно определение.

(4) *Конфигурацией* называется выражение, удовлетворяющее следующим условиям:

. оно не содержит вхождений символов $П$ и $Л$;

.. оно содержит ровно одно вхождение символа типа q_i;

... это вхождение символа q_i не занимает в выражении крайней правой позиции.

Так, выражения

$$1q_3S_4BS_2BBS_1$$

и

$$q_3B1$$

являются конфигурациями, а выражения

$$Лq_1, \quad 1S_2, \quad q_2BS_3q_3S_1, \quad S_1S_2q_1$$

таковыми не являются.

Пусть α — некоторая конфигурация. Будем говорить, что α есть конфигурация некоторой машины Тьюринга Z, если состояние q_i, фигурирующее в α, является одним из внутренних состояний машины Z, а все остальные символы, входящие в α, принадлежат алфавиту Z.

Конфигурация машины однозначно задает ее поведение[1]). В самом деле, характеристика ситуации не учитывает всю совокупность символов, записанных на ленте; конфигурация же позволяет сделать это.

Конфигурация, из которой выброшено q_i, характеризует содержимое ленты; в то же время, если S_j — символ, стоящий в конфигурации справа от q_i, то q_iS_j есть ситуация машины Z.

Дадим теперь последнее определение.

(5) Пусть α и β — конфигурации некоторой машины Z. Мы будем писать

$$\alpha \to \beta$$

в одном из трех следующих случаев (P и Q обозначают произвольные последовательности символов алфавита машины):

$\exists P, Q: \alpha = Pq_iS_jQ, \quad \beta = Pq_lS_kQ,$ машина имеет команду $q_iS_jS_kq_l$,
$\exists P, Q: \alpha = Pq_iS_jQ, \quad \beta = PS_jq_lQ,$ машина имеет команду $q_iS_jЛq_l$,
$\exists P, Q: \alpha = PS_kq_iS_jQ, \beta = Pq_lS_kS_jQ,$ машина имеет команду $q_iS_iПq_l$.

. Первый случай соответствует такому элементарному действию машины, при котором она заменяет символ S_j символом S_k и переходит из состояния q_i в состояние q_l.

.. Второй случай соответствует действию, состоящему в передвижении ленты на одну ячейку влево и переходе из состояния q_i в состояние q_l.

... Третий случай соответствует передвижению ленты на одну ячейку вправо и переходу из состояния q_i в состояние q_l.

Из определения (5) непосредственно следует, что

. если $\alpha \to \beta$ и $\alpha \to \gamma$, то $\beta = \gamma$[2]);

.. если Z и Z' — две машины Тьюринга, такие, что всякая команда машины Z есть команда машины Z' (в этом случае пишут $Z \subset Z'$), то из $\alpha \to \beta$ относительно Z следует $\alpha \to \beta$ относительно Z'.

Вычисления. Конфигурация α называется *заключительной* относительно машины Z, если не существует конфигурации β этой машины, такой, что $\alpha \to \beta$.

---

[1]) Чтобы это утверждение имело смысл, конфигурации, определенные абстрактным образом как цепочки некоторого специального вида, должны быть интерпретированы в терминах «частей» машины.

Здесь имеется в виду следующая интерпретация. Прежде всего предполагается, что в каждый момент лишь конечное число ячеек непусто (иначе говоря, занято символами, отличными от B). Тогда цепочка, получаемая из конфигурации вычеркиванием символа q_i, интерпретируется как последовательность символов, записанных подряд, слева направо, в некотором куске ленты, содержащем все непустые (в данный момент) ячейки; q_i означает состояние машины в данный момент; вхождение символа, перед которым стоит q_i, — то, которое в данный момент читается. — *Прим. ред.*

[2]) Здесь используется *детерминированность* машины, т. е. требование различия начальных пар символов в различных командах. — *Прим. ред.*

Назовем *вычислением машины Тьюринга* Z последовательность конфигураций

$$\alpha_1, \alpha_2, \ldots, \alpha_p,$$

таких, что $\alpha_i \to \alpha_{i+1}$ $(1 \leqslant i < p)$ и α_p — заключительная конфигурация (относительно Z). Будем писать

$$\alpha_p = \text{Рез}\, Z(\alpha_1),$$

что означает: «α_p есть результат переработки α_1 машиной Z».

Начальная конфигурация. Условимся считать, что в общем случае машина перерабатывает слово, записанное в алфавите $\{S_i\}$; при этом S_0 обозначает пустой символ и исходное слово окружено пустыми символами с обеих сторон. Машина располагает таким количеством пустых ячеек, какое необходимо ей для вычислений. Одно из состояний машины принимается за *начальное*; машина начинает работу, находясь в этом состоянии, причем головка установлена перед самым левым (непустым) символом, записанным на ленте.

4.2.3. Примеры машин Тьюринга

Пример 1. Пусть машина Z имеет алфавит $\mathfrak{A} = \{S_0, S_1, S_2\}$ и состояния q_1, q_2, q_3; состояние q_1 является начальным. Поведение машины описывается следующими командами:

$$q_1\ S_0\ Л\ q_1 \qquad (1)$$
$$q_1\ S_2\ Л\ q_1 \qquad (2)$$
$$q_1\ S_1\ Л\ q_2 \qquad (3)$$
$$q_2\ S_1\ П\ q_3 \qquad (4)$$
$$q_2\ S_0\ Л\ q_1 \qquad (5)$$
$$q_2\ S_2\ Л\ q_1 \qquad (6)$$
$$q_3\ S_1\ S_2\ q_3 \qquad (7)$$

Пояснения. а) Если машина находится в начальном состоянии q_1 и читает символ S_0 или S_2, то она сдвигает ленту на одну ячейку влево, оставаясь в состоянии q_1; теперь головка обозревает символ, стоящий в ячейке непосредственно справа от S_0 или S_2 (команды (1) и (2)).

б) Если машина находится в состоянии q_1 и читает символ S_1, то она сдвигает ленту на одну ячейку влево и в результате начинает обозревать символ, соседний справа к S_1. При этом она «запоминает» тот факт, что видела S_1: состояние q_2 можно интерпретировать как информацию о том, что был прочитан символ S_1 (команда (3)).

в) Если, все еще «помня об S_1», машина видит S_0 или S_2, она забывает, что видела S_1 (возврат в состояние q_1), и переходит к

обозреванию символа, стоящего непосредственно справа от S_0 или S_2 (команды (5) и (6)).

г) Если, напротив, находясь в состоянии q_2 («помня об S_1»), машина снова видит S_1, она возвращается к предыдущему символу (это как раз тот S_1, который привел ее в состояние q_2) и переходит в состояние q_3 (команда (4)). Затем она заменяет первый из этих двух соседних символов S_1 на S_2 и переходит в состояние q_3 (команда (7)). Теперь машина оказывается в ситуации q_3S_2, для которой нет подходящей команды; поэтому она останавливается.

д) Таким образом, единственным фактором, влекущим остановку машины, является наличие двух последовательно написанных S_1. Если в процессе чтения входного слова машина не встретит двух рядом стоящих S_1, то она дойдет до конца слова, перейдет на пустые ячейки справа от него и будет работать вечно (команда (1)).

Зададим машине Z, находящейся в состоянии q_1, слово

$$S_1S_0S_2S_1S_1S_0S_1.$$

В результате получится последовательность конфигураций:

$q_1\ S_1\ \ S_0\ \ S_2\ \ S_1\ \ S_1\ S_0\ S_1$ (1)

$S_1\ q_2\ S_0\ \ S_2\ \ S_1\ \ S_1\ S_0\ S_1$ (2)

$S_1\ \ S_0\ q_1\ S_2\ \ S_1\ \ S_1\ S_0\ S_1$ (3)

$S_1\ \ S_0\ \ S_2\ q_1\ S_1\ \ S_1\ S_0\ S_1$ (4)

$S_1\ \ S_0\ \ S_2\ \ S_1\ q_2\ S_1\ S_0\ S_1$ (5)

$S_1\ \ S_0\ \ S_2\ q_3\ S_1\ \ S_1\ S_0\ S_1$ (6)

$S_1\ \ S_0\ \ S_2\ q_3\ S_2\ \ S_1\ S_0\ S_1$ (7) (заключительная конфигурация)

Теперь предложим той же машине слово

$$S_1S_2S_2.$$

В этом случае вычисления пойдут так:

$q_1\ S_1\ \ S_2\ \ S_2$ (1)

$S_1\ q_2\ S_2\ \ S_2$ (2)

$S_1\ \ S_2\ q_1\ S_2$ (3)

$S_1\ \ S_2\ \ S_2\ q_1\ S_0$ (4)

$S_1\ \ S_2\ \ S_2\ \ S_0\ q_1\ S_0$ (5)

Процесс продолжается вечно, т. е. результат здесь не определен.

Пример 2. *Транскрибирующая машина.* На ленте машины Тьюринга записано слово в алфавите $\mathfrak{A} = \{a_1, \ldots, a_n\}$; слева и справа его окружают пустые ячейки #.

Пусть имеется другой алфавит $\mathfrak{B} = \{b_1, \ldots, b_n\}$, находящийся с первым во взаимно однозначном соответствии: $a_i \leftrightarrow b_i$.

Задача машины T состоит в транскрибировании любого данного $\mathfrak{A}$-слова с помощью алфавита $\mathfrak{B}$, причем транскрибированное слово должно быть записано на другом участке ленты, а исходное сохранено.

В процессе работы машина использует маркер m. Вот совокупность команд, или *программа*, определяющая машину T.

I. Для всякого i:

I. 1. $\quad q_0\ a_i\ m\ q_i.$

Находясь в начальном состоянии q_0 и прочитав a_i — самую правую букву исходного слова, машина пишет вместо a_i маркер m и запоминает a_i посредством перехода в состояние q_i.

I. 2. $\quad q_i\ m\ П\ q_i.$

Находясь в состоянии q_i (т. е. помня о символе a_i) и видя маркер m, T перемещает ленту вправо, оставаясь в состоянии q_i.

I. 3. $\quad q_i\ \#\ a_i\ q_i^i.$

Находясь в состоянии q_i и видя пустую ячейку, машина T записывает в нее a_i и переходит в состояние q_i^i: «прочитанный перед этим символ a_i скопирован».

I. 4. $\quad q_i^i\ m\ Л\ q_i^i.$

Видя маркер после копирования a_i (**не обязательно сразу**), машина сдвигает ленту влево.

II. Для всякой пары $\langle i, j\rangle$:

II. 1. $\quad q_i\ a_j\ Л\ q_i.$

II. 2. $\quad q_i^i\ a_j\ Л\ q_i^i.$

II. 3. $\quad q_i^i\ b_j\ Л\ q_i^i.$

Скопировав a_i и находясь в состоянии q_i^i, машина вне зависимости от обозреваемой буквы передвигает ленту влево; при этом она остается в состоянии q_i^i.

II. 4. $\quad q_i^i\ \#\ b_i\ r.$

Дойдя до пустой ячейки, T транскрибирует a_i, вписывая b_i в пустую ячейку, после чего переходит в состояние r — «состояние возврата».

III. Для всякого j:

III. 1. $\quad r\ b_j\ П\ r.$

III. 2. $\quad r\ a_j\ П\ r.$

Когда машина находится в состоянии r, то, какова бы ни была обозреваемая буква, T остается в состоянии r и сдвигает ленту вправо.

III. 3. $r\ m \Rightarrow r.$

T продолжает сдвигать ленту вправо до тех пор, пока не наткнется на маркер, оставленный ею там, откуда она «отправилась» искать свободные места для копирования и транскрибирования a_i. Тогда она стирает маркер, оставаясь в состоянии r.

III. 4. $r \Rightarrow Л\ q_0.$

Видя пустой символ, которым она только что заменила маркер, машина сдвигает ленту влево (так что теперь она будет обозревать очередной символ исходного слова) и возвращается в начальное состояние q_0.

Когда T закончит транскрибирование, она окажется в состоянии q_0, но сможет прочесть только какую-либо из букв b_j. Поскольку ситуации $\langle q_0, b_j\rangle$ не соответствует ни одна команда, машина остановится.

В этот момент на ленте будут записаны оба слова (исходное $\mathfrak{A}$-слово и его $\mathfrak{B}$-транскрипция), разделенные пустой ячейкой.

§ 4.3. «ЧИСЛЕННЫЕ» МАШИНЫ ТЬЮРИНГА

Описанные в п. 4.2.2 машины Тьюринга позволяют реализовать алгоритмы самого общего характера. В частности, два предшествующих примера относились к «нечисленным» алгоритмам. Однако, по правде говоря, на уровне машин Тьюринга между «численным» и «нечисленным» нет существенного различия[1]). Когда речь идет просто о манипуляциях с символами, мы вольны давать этим символам любую интерпретацию — и числовую, и нечисловую. Если мы хотим проводить вычисления, т. е. работать с числами, то их можно изображать многими разными способами.

Для работы с целыми неотрицательными числами мы будем пользоваться абстрактным двубуквенным алфавитом $\mathfrak{A} = \{S_0, S_1\}$:

S_0 соответствует пустому символу, обозначаемому B;

S_1 соответствует палочке, обозначаемой |.

Всякое целое неотрицательное число n изображается последовательностью из $n + 1$ палочек. Это соглашение позволяет отличать число «нуль», изображаемое палочкой, от пустого символа,

[1]) Для реальных вычислительных машин это различие, напротив, весьма важно (с практической точки зрения). Пригодность машины для той или иной области приложений зависит от набора «встроенных» операций: машины для количественных расчетов нуждаются в разнообразных арифметических операциях, машины для логической обработки информации — в булевых операциях, операциях типа подстановки и т. д.

играющего роль разделителя. Мы будем обозначать подобное изображение числа n через $\bar{n}$; например:

$$\bar{0} = |,\ \bar{1} = ||,\ \ldots,\ \bar{4} = |||||.$$

Конечная последовательность $n_1, n_2, \ldots, n_k$ будет изображаться следующим образом:

$$\bar{n}_1 B \bar{n}_2 B \ldots B \bar{n}_k.$$

Пусть имеется выражение M, иначе говоря, цепочка в алфавите $\{B, |; q_1, q_2, \ldots, q_n; П, Л\}$. Будем обозначать через $\langle M \rangle$ число палочек, которое она содержит.

4.3.1. Понятие вычислимой функции

Пусть Z — машина Тьюринга с состояниями $q_1, q_2, \ldots$. Каждой n-ке $(m_1, \ldots, m_n)$ неотрицательных целых чисел мы поставим в соответствие конфигурацию

$$\alpha_1 = q_1 \bar{m}_1 B \bar{m}_2 B \ldots B \bar{m}_n.$$

Если для машины Z существует вычисление, начинающееся конфигурацией α_1 и доходящее до заключительной конфигурации α_p:

$$\alpha_1 \to \alpha_2 \to \ldots \to \alpha_p,$$

то число $\langle \alpha_p \rangle$ есть функция от Z и начальной n-ки; мы будем писать

$$\langle \alpha_p \rangle = \Psi_Z^{(n)}(m_1, \ldots, m_n).$$

Если же вычисления, начинающегося с α_1, не существует, т. е. не существует целого числа p, такого, что конфигурация α_p является заключительной, то функция $\Psi_Z^{(n)}$ не определена для рассматриваемой n-ки.

Исходя из машины Z, мы пришли к функции с целыми неотрицательными значениями, определенной на множестве N^n или, быть может, на некотором подмножестве множества N^n (частично определенная функция) [1]).

Поступим теперь наоборот: попробуем исходить из функций, определенных на N^n или его подмножестве.

Определение. Будем говорить, что функция f, определенная на некотором подмножестве множества N^n, является *частично вычислимой*, если существует машина Тьюринга Z, такая, что для всякой n-ки $(x_1, \ldots, x_n)$, которой отвечает некоторое значение f, выполняется равенство

$$f(x_1, \ldots, x_n) = \Psi_Z^{(n)}(x_1, \ldots, x_n).$$

[1]) Здесь и далее N означает натуральный ряд (включающий нуль), а N^n — множество всевозможных упорядоченных n-ок натуральных чисел. — *Прим. ред.*

Назовем функцию f *вычислимой*, если она определена на N^n и является частично вычислимой [1]).

4.3.2. Пример вычислимой функции

Функция $f(x, y) = x + y$, определенная на множестве пар неотрицательных целых чисел, является «вычислимой в интуитивном смысле»; покажем, что она вычислима и в смысле Тьюринга. Для этого построим машину Тьюринга Z с алфавитом $\{B, |\}$, такую, что

$$x + y = \Psi_Z^{(2)}(x, y).$$

Такая машина должна, имея на ленте коды $\bar{x}$ и $\bar{y}$, разделенные пустой ячейкой, получать из этой записи $x + y$ палочек (не обязательно подряд). Заметим, что

$$x + y = \langle \bar{x} \rangle + \langle \bar{y} \rangle - 2.$$

Читатель без труда убедится, что следующая машина выполняет поставленную задачу:

$q_1 \mid B\ q_2$	(1)	Стирается первая палочка первого кода.
$q_2\ B\ Л\ q_3$	(2)	Лента сдвигается до тех пор, пока
$q_3 \mid Л\ q_3$	(3)	головка не дойдет до первой палочки
$q_3\ B\ Л\ q_4$	(4)	второго кода.
$q_4 \mid B\ q_4$	(5)	Стирается первая палочка второго кода.

Здесь машина останавливается, поскольку ситуации q_4B нет ни в одной команде. В этот момент на ленте ровно $x + y$ палочек.

4.3.3. Пример частично вычислимой функции

Функция $g(x, y) = x - y$ определена лишь для тех пар, у которых $x \geqslant y$; покажем, что она частично вычислима в смысле Тьюринга.

Для этого мы построим машину Тьюринга Z, выполняющую вычитание. Пусть на ленте имеются коды $\bar{x}$ и $\bar{y}$, разделенные пустой ячейкой: $\bar{x}$ слева, $\bar{y}$ справа. Машина будет работать последовательными циклами; каждый цикл состоит в стирании одной (самой левой) палочки в $\bar{x}$ и одной (самой правой) палочки в $\bar{y}$. Вот соответствующая программа.

$q_1 \mid B\ q_1$	(1)	Стирается самая левая палочка в $\bar{x}$.
$q_1\ B\ Л\ q_2$	(2)	Лента сдвигается на одну ячейку влево.

---

[1]) Более принята другая терминология: частичную функцию $f(x_1, \ldots, x_n)$ называют *вычислимой*, если существует машина Тьюринга Z, такая, что для всякой n-ки $x_1, \ldots, x_n$, которой отвечает значение f, выполняется равенство

$$f(x_1, \ldots, x_n) = \Psi_Z^{(n)}(x_1, \ldots, x_n),$$

а для тех n-ок, для которых значения f не существуют, машина работает вечно. — *Прим. ред.*

$q_2 \mid Л\ q_2$ (3) Лента сдвигается влево до тех пор, пока головка не дойдет до конца кода $\bar{x}$.

$q_2\ B\ Л\ q_3$ (4) Головка переходит через пробел (пустую ячейку), разделяющий $\bar{x}$ и $\bar{y}$.

$q_3 \mid Л\ q_3$ (5) Лента сдвигается влево до тех пор, пока головка не дойдет до конца кода $\bar{y}$.

$q_3\ B\ П\ q_4$ (6) Попав на пустую ячейку, примыкающую к $\bar{y}$ справа, головка возвращается на одну ячейку назад (т. е. к самой правой палочке $\bar{y}$).

$q_4 \mid B\ q_4$ (7) Самая правая палочка в $\bar{y}$ стирается.

$q_4\ B\ П\ q_5$ (8) Лента сдвигается на одну ячейку вправо и машина переходит в «проверочное» состояние.

Дело в том, что продолжение или прекращение вычисления зависит от результатов некоторых проверок.

$q_5 \mid П\ q_6$ (9) Если обозреваемая ячейка содержит палочку ($\bar{y}$ еще не стерт полностью), вычисление продолжается. Если же обозреваемая ячейка пуста, то, поскольку ситуации q_5B не отвечает никакая команда, машина останавливается.

$q_6 \mid П\ q_6$ (10) Лента сдвигается вправо до тех пор, пока головка не дойдет до начала кода $\bar{y}$.

$q_6\ B\ П\ q_7$ (11) Головка переходит через пробел, разделяющий $\bar{y}$ и $\bar{x}$.

$q_7 \mid П\ q_8$ (12) Если в $\bar{x}$ осталась хотя бы одна палочка, вычисление продолжается.

$q_7\ B\ П\ q_7$ (13) Если в $\bar{x}$ не осталось ни одной палочки (случай $x < y$), машина будет работать вечно, сдвигая ленту вправо.

$q_8 \mid П\ q_8$ (14) Лента сдвигается вправо до тех пор, пока головка не дойдет до начала того, что осталось от кода $\bar{x}$.

$q_8\ B\ Л\ q_1$ (15) Головка устанавливается так, что обозревается самая левая палочка «остатка» кода $\bar{x}$; машина переходит в начальное состояние q_1. Тем самым она оказывается готовой начать очередной цикл.

Частично вычислимую функцию $x - y$ нетрудно тривиальным образом продолжить до вычислимой функции (это делается ниже, в упр. 4.4.8). Из этого, разумеется, нельзя делать вывод, что всякую частично вычислимую функцию можно продолжить до вычислимой.

§ 4.4. УПРАЖНЕНИЯ

1. Построить машину Тьюринга, которая копировала бы заданное слово, оставляя между соседними буквами пустые ячейки (т. е. переписывала бы его «вразрядку»).

2. Построить машину Тьюринга, которая окаймляла бы маркерами заданное слово.

3. Построить машину Тьюринга, которая писала бы зеркальный образ заданного слова.

4. Машину Тьюринга можно определить посредством множества пятерок вида

$$\langle q_i,\ S_j,\ S_k,\ q_l,\ \delta_m\rangle,$$

таких, что если машина, находясь в состоянии q_i, обозревает символ S_j, то она пишет символ S_k, переходит в состояние q_l и перемещает головку способом δ_m, где δ_m принимает одно из трех значений:

-1 : сдвиг влево;

0 : отсутствие сдвига;

$+1$: сдвиг вправо.

Переформулировать изложенные выше фрагменты теории машин Тьюринга и конкретные примеры тьюринговых вычислений, используя такой вариант определения машины.

5. Построить машину Тьюринга, которая вычисляла бы следующую функцию:

$$\mathrm{suc}(x) = x + 1.$$

6. Выписать вычисления, реализующие вычитания 3—2 и 2—3 в машине, описанной в п. 4.3.3.

7. Построить машину Тьюринга, которая вычисляла бы i-ю проекцию n-ки натуральных чисел:

$$U_i^{(n)}(x_1,\ x_2,\ \ldots,\ x_n) = x_i,\ 1 \leqslant i \leqslant n.$$

8. Построить машину Тьюринга, которая вычисляла бы функцию $x \dot{-} y$, определенную следующим образом:

$$x \dot{-} y = x - y, \text{ если } x \geqslant y;$$

$$x \dot{-} y = 0, \qquad \text{если } x < y.$$

9. Построить машину Тьюринга, которая вычисляла бы ту же функцию $x \dot{-} y$, но при этом на каждом шаге вычисления окаймляла непустую часть ленты маркерами.

10. а) Построить машину Тьюринга, которая вычисляла бы произведение $(x + 1)(y + 1)$. б) Изменить эту машину так, чтобы она на каждом шаге вычисления окаймляла непустую часть ленты маркерами.

Глава V

ВЫЧИСЛИМОСТЬ И РАЗРЕШИМОСТЬ

В настоящей главе читатель найдет некоторые понятия, связанные с вычислимостью и разрешимостью; эти понятия оказываются необходимыми, поскольку они связаны с теорией формальных грамматик.

Во многих случаях, чтобы не перегружать изложение и лучше выделить основную линию рассуждений, мы вынесли в упражнения технические детали доказательств.

§ 5.1. ИСЧИСЛЕНИЕ ФУНКЦИЙ

Пример с вычитанием (п. 4.3.3), по-видимому, ясно показывает, что для вычисления даже простых функций могут понадобиться очень сложные машины Тьюринга. Чтобы избежать необходимости строить такие машины непосредственно, можно воспользоваться композицией элементарных стандартных машин подобно тому, как программа для ЭВМ составляется из элементарных программ.

5.1.1. Сочленение и стандартизация

Для достижения только что указанной цели мы будем использовать следующие средства.

Различение состояний. Когда некоторая машина Тьюринга строится из нескольких готовых машин, нужно уметь отличать состояния результирующей машины от состояний составляющих ее машин (каждая из них может иметь, например, состояние q_1). Для этого мы примем следующее соглашение.

Если Z — машина Тьюринга, то через $Z^{(n)}$ обозначается аналогичная машина, все состояния которой получаются из состояний машины Z путем увеличения их индексов на n. (Таким образом, мы переходим от Z к $Z^{(n)}$ с помощью точно такой же операции, которая используется в программировании для ЭВМ: в машину вводится подпрограмма, первая команда которой помещается в ячейку с адресом n.)

Стандартная машина. При композиции машин Тьюринга необходимо, чтобы результаты вычислений, полученных одной машиной, могли служить исходными данными для другой машины. Для этого целесообразно ввести так называемое *стоп-состояние.*

Мы уже приняли соглашение, что любая машина начинает любое вычисление в состоянии q_1. Аналогично будем предполагать, что любая машина заканчивает вычисления в таком состоянии q_θ, что θ есть наибольший из индексов, приписываемых символам внутренних состояний машины, и при этом ни одна из ситуаций машины Z не допускает состояния q_θ.

Машина, снабженная стоп-состоянием, является по определению *стандартной машиной.*

Чистка. Напомним, что результат тьюрингова вычисления — это число палочек, оказавшихся на ленте машины в конце ее работы. Однако алфавит машины может содержать символы, отличные от пробела и палочки. Поэтому, чтобы результат работы одной машины мог служить «входным заданием» для другой, необходимо сгруппировать все палочки вместе, а затем стереть все прочие символы.

Маркировка. Когда машина Тьюринга «изучает» свои исходные данные (которые могут быть результатом работы предыдущей машины), возникает опасность бесконечного протягивания ленты. Чтобы избежать этого, всякая стандартная машина Z преобразуется в другую стандартную машину Z', которая выполняет те же вычисления, что и Z, но, кроме того, обрамляет любой результат двумя специальными маркерами — левым и правым. Построению подобных машин для отдельных частных случаев были посвящены некоторые упражнения в конце предыдущей главы. Общий случай рассматривается в упр. 5.1.2.

Сохранение промежуточных результатов. Поскольку при композиции машин Тьюринга они не всегда сочленяются линейно, может оказаться необходимым сохранять в ходе вычислений некоторые промежуточные результаты. Подобное запоминание также может быть выполнено с помощью маркеров.

Более точно, если машина Z ставит в соответствие заданию $(d_1, \ldots, d_i)$ результат $(r_1, \ldots, r_j)$, то можно построить такую машину Z_1, которая ставила бы в соответствие заданию $(p_1, \ldots, p_k, d_1, \ldots, d_i)$ результат $(p_1, \ldots, p_k, r_1, \ldots, r_j)$.

В частности, может оказаться необходимым сохранить исходное задание. В таком случае его можно скопировать в какой-либо части ленты (например, с помощью машины, выполняющей транскрибирование) и сохранять его там с помощью некоторой машины Z_1 указанного в предыдущем абзаце типа.

5.1.2. Упражнение. Машина с маркерами

Пусть имеется стандартная машина Z с алфавитом

$$\{S_0 \text{ (или } B), S_1 \text{ (или } |), S_2, \ldots\}$$

и состояниями $q_1, q_2, \ldots, q_\theta$ Требуется построить машину Z', которая выполняла бы те же вычисления, что и Z, но обрамляла бы результат маркерами γ слева и δ справа (если какой-нибудь из этих символов уже есть в алфавите машины Z, то его, разумеется, необходимо заменить).

Указания:

а) Построить машину Z_1, которая читает исходное задание, находит его концы по наличию двух пустых ячеек подряд, ставит маркеры γ и δ, а затем перемещает ленту так, чтобы головка обозревала γ.

Чтобы определить Z_1, достаточно пяти состояний.

б) Построить машину Z_2, добавив к состояниям машины Z_1 пять новых состояний, а к ее программе такие команды, в соответствии с которыми Z_2 будет в случае необходимости перемещать γ (или δ) на одну ячейку влево (соответственно вправо) и возвращаться всякий раз к основному вычислению.

в) Построить стандартную машину Z_3, соответствующую машине Z_2.

г) Построить машину Z_4, которая стирала бы на ленте все символы, кроме палочек, собирала бы все палочки вместе и добавляла к ним еще одну (чтобы результат можно было понимать как число).

5.1.3. Упражнение

а) Построить машину Тьюринга, которая кодировала бы алфавит $\{a, b, c\}$ с помощью алфавита $\{0, 1\}$ по образцу, описанному в предыдущей главе.

б) Построить машину, которая расшифровывала бы указанную кодировку, а для нерегулярных слов (не являющихся кодами) распознавала бы их нерегулярность.

§ 5.2. ОПЕРАЦИИ НАД ФУНКЦИЯМИ

Вернемся теперь к функциям, отображающим N^n в N.

5.2.1. Композиция

Пусть имеется, с одной стороны, функция от m переменных

$$f(x_1, \ldots, x_m),$$

определенная на $D \subset N^m$, с другой стороны, m функций от n переменных

$$g_1(y_1, \ldots, y_n),$$
$$\ldots\ldots\ldots$$
$$g_m(y_1, \ldots, y_n),$$

где каждая g_i имеет область определения $D_i \subset N^n$.

Операция *композиции* сопоставляет этим функциям новую функцию

$$h(y_1, \ldots, y_n) = f[g_1(y_1, \ldots, y_n), \ldots, g_m(y_1, \ldots, y_n)].$$

Областью определения функции h является подмножество пересечения $D_1 \cap \ldots \cap D_m$, состоящее из всех n-ок, для которых значения функций $g_1, \ldots, g_m$ дают точку из D.

Основным результатом, относящимся к композиции вычислимых функций, является следующая

Теорема. *Если функции f, $g_1, \ldots, g_m$ частично вычислимы (соответственно вычислимы), то функция h также частично вычислима (соответственно вычислима).*

Доказательство этого предложения проводится с помощью композиции стандартных машин Тьюринга, соответствующих функциям f, $g_1, \ldots, g_m$. Само по себе доказательство не сложно, и мы предоставляем его читателю в качестве упражнения.

Из данной теоремы непосредственно следует, что классы вычислимых и частично вычислимых функций замкнуты относительно композиции.

5.2.2. Минимизация

Операция *минимизации* сопоставляет функции $f(y, x_1, \ldots, x_n)$, определенной всюду на N^{n+1}, функцию $h(x_1, \ldots, x_n)$, значение которой для любой данной n-ки равно наименьшему значению y, такому, что $f(y, x_1, \ldots, x_n) = 0$, если только такие значения y существуют; если же для данных $x_1, \ldots, x_n$ функция $f(y, x_1, \ldots, x_n)$ не обращается в нуль ни при каких значениях y, то $h(x_1, \ldots, x_n)$ для данных $x_1, \ldots, x_n$ не определена.

Функция h, таким образом, может оказаться определенной лишь на некотором подмножестве множества N^n; если h определена на всем множестве N^n, то функция $f(y, x_1, \ldots, x_n)$, из которой h получается минимизацией, называется *регулярной*.

Основным результатом, относящимся к минимизации, является следующая

Теорема. *Если функция $f(y, x_1, \ldots, x_n)$ вычислима, то функция h, полученная из нее минимизацией, частично вычислима. Если при этом функция f регулярна, то функция h вычислима.*

Эта теорема доказывается путем построения машины Тьюринга, которая вычисляет последовательно $f(0, x_1, \ldots, x_n)$, $f(1, x_1, \ldots, x_n), \ldots$ до тех пор, пока для f не будет получено значение нуль (если функция f никогда не обращается в нуль, машина будет продолжать вычисления вечно). Здесь опять следует использовать стандартные машины Тьюринга.

5.2.3. Упражнения

1. Доказать теорему о композиции функций.

2. Минимизация:

а) Построить стандартную машину Тьюринга T, которая каждой n-ке $(x_1, \ldots, x_n)$ сопоставляла бы $(n+1)$-ку $(0, x_1, \ldots, x_n)$.

б) Пусть $f(y, x_1, \ldots, x_n)$ — (частично) вычислимая функция. Доказать, что существует стандартная машина U, которая сопоставляет любым заданным $n+1$ числам $y, x_1, \ldots, x_n$ представление системы чисел $[f(y, x_1, \ldots, x_n), y, x_1, \ldots, x_n]$, если только $f(y, x_1, \ldots, x_n)$ существует.

Увеличить индексы состояний машины U так, чтобы ее можно было присоединить к T.

в) Присоединить к U машину V, уменьшающую число палочек в представлении числа $f(y, x_1, \ldots, x_n) = r$ на единицу для того, чтобы можно было распознать, равно ли это число нулю.

г) Присоединить к построенным машинам еще одну машину W, которая в случае $r \neq 0$ прибавляла бы к $\overline{y}$ палочку и переходила бы тем самым к $\overline{y+1}$.

д) Присоединить к соединению построенных машин стандартную машину Y, которая при $r = 0$ очищала бы ленту от символов, отличных от $|$, и делала содержимое ленты равным y.

е) Убедиться, что машина, представляющая собой соединение машин T, U, V, W и Y, обладает всеми нужными свойствами.

3. Напомним, что функции $(x+1)(y+1)$ и $x \dot{-} y$ вычислимы; доказательство этого предлагалось выше (см. § 4.4) в качестве упражнения. (Функция $x \dot{-} y$ получается тривиальным продолжением функции $x - y$, а именно $x - y$ имеет значение либо $x - y$, либо нуль в зависимости от того, определено $x - y$ или нет.)

Опираясь на вычислимость этих функций (а также на вычисление проекций, см. § 4.4), доказать, что функция xy также вычислима.

4. Доказать, что функция x^h вычислима для любого $h \in N$.

5.2.4. Модифицированные машины Тьюринга

Можно рассмотреть различные расширения понятия машины Тьюринга. Например:

. Можно присоединить к машине Z некоторое конечное множество натуральных чисел A и множество команд $S_i q_j q_k q_l$, означающих, что в ситуации (S_i, q_j) и для конфигурации α машина Z переходит либо в состояние q_k, либо в состояние q_l в зависимости от того, принадлежит ли содержимое $\langle\alpha\rangle$ данной конфигурации множеству A. Такое расширение машин Тьюринга, рассмотренное Дэвисом, не увеличивает их вычислительную мощность.

.. Можно добавить к машине Тьюринга разного рода специализированные компоненты: ленту для входных данных, ленту для окончательных результатов, ленту для промежуточных вычисле-

ний, более сложные команды и т. п. Вообще говоря, нетрудно показать, что такие сложные машины эквивалентны простым машинам.

§ 5.3. МЕТОД ГЁДЕЛЯ

Мы уже имели случай заметить, что на уровне машин Тьюринга различение «численных» и «нечисленных» операций не имеет разумного оправдания.

Еще в начале 30-х годов Курт Гёдель применил числовое кодирование имен некоторых метаматематических объектов в доказательстве своей знаменитой теоремы о неполноте арифметики.

Метод Гёделя позволяет оперировать с нечисленными объектами, как с числами; программисты, для которых действия над адресами — дело привычное, сразу же поймут этот подход.

Итак, пусть имеется некоторый алфавит и цепочки из символов этого алфавита.

Каждой такой цепочке, представляющей собой нечисленный объект, мы поставим в соответствие некоторое целое положительное число — ее *гёделевский номер*, обладающий следующими свойствами:

. Существует машина Тьюринга, которая для любой цепочки, написанной на ее ленте, выдает в качестве окончательного результата ее гёделевский номер.

.. Существует машина Тьюринга, которая для любого числа, написанного на ее ленте, выдает один из двух следующих ответов:

— либо «Не существует цепочки, соответствующей этому числу» (т. е. цепочки, для которой данное число являлось бы ее гёделевским номером);

— либо «Цепочка, соответствующая этому числу, есть...».

Таким образом, гёделевский номер, сопоставленный цепочке, полностью определяет ее; он является ее *собственным именем* и одновременно, будучи числом, может участвовать в вычислениях.

Аналогичным образом можно нумеровать *последовательности цепочек*.

5.3.1. Один из возможных способов нумерации цепочек

Говоря о машинах Тьюринга, мы используем выражения, в которых фигурируют:

символы S и q с различными индексами;

символы $Л$ и $П$.

Для записи индексов мы будем пользоваться двоичной системой счисления, причем роль цифр будут играть символы z(*zéro*, т. е. 'нуль') и u(*un*, т. е. 'один'); будем писать индексы в строчку:

S_0 записывается как Sz,

S_1 записывается как Su,

S_2 записывается как *Suz*,
S_3 записывается как *Suu*,
q_5 записывается как *quzu*.

Если нам понадобятся последовательности цепочек, мы будем пользоваться разделительным знаком «!».

Итак, мы имеем следующие символы:

$$\{S,\ q,\ z,\ u,\ Л,\ П,\ !\}.$$

Занумеруем цепочки, состоящие из этих символов, следующим образом.

Пустой цепочке сопоставим число 0.

Цепочкам длины 1 сопоставим соответственно

$$S,\ q,\ z,\ u,\ Л,\ П,\ !,$$
$$1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7.$$

Произвольной цепочке сопоставляется число (в восьмеричной системе счисления при указанном выше кодировании цифр), которое является ее каноническим представлением.

Пример. Команда $q_3S_5S_0q_4$ записывается как

$$quuSuzuSzquzz;$$

этому выражению соответствует восьмеричное число 2441434132433.

Мы видим, что номер цепочки определяется по ней чисто механически; стало быть, это может делать машина Тьюринга.

Если мы пойдем в обратном направлении, т. е. будем отправляться «от чисел», то увидим, что

числу 0 соответствует пустая цепочка;

восьмеричному числу, не содержащему цифры 0, соответствует единственная цепочка или единственная последовательность цепочек;

восьмеричному числу, содержащему хотя бы одно вхождение цифры 0, не соответствует ни одной цепочки или последовательности цепочек.

5.3.2. Гёделевский номер машины Тьюринга

Машина Тьюринга может рассматриваться как множество команд $Q_1, Q_2, \ldots, Q_n$.

Сопоставим каждой команде ее гёделевский номер

$$\mathrm{ng}(Q_1),\ \mathrm{ng}(Q_2),\ \ldots,\ \mathrm{ng}(Q_n),$$

а затем расположим эти номера в порядке возрастания. (Нам даже не придется использовать разделитель $7 = !$, поскольку $1 = S$ и $2 = q$ служат маркерами.)

Таким образом каждая машина Тьюринга получит свой гёделевский номер. Поскольку процедура приписывания номеров

машинам является механической, мы можем поручить ее выполнение некоторой машине Тьюринга, которая будет обозначаться через G_1; любопытно отметить, что G_1 вычисляет, в частности, и свой собственный гёделевский номер.

Обратно, существует такая машина Тьюринга G_2, которая для любого восьмеричного числа выдает один из двух следующих ответов:

— либо «Это число не является гёделевским номером никакой машины Тьюринга»;

— либо «Это число является гёделевским номером машины Тьюринга, имеющей следующие команды...».

Заметим, что если G_2 получает на вход свой собственный гёделевский номер, то она выдает ответ второго типа.

5.3.3. Упражнение: другой способ кодирования

В оригинальных работах Гёделя использовался следующий способ нумерации. Пусть требуется закодировать выражение

$$M = `\gamma_{i_1}\gamma_{i_2}\dots\gamma_{i_n}',$$

которое составлено из символов, принадлежащих набору $\{\gamma_1, \gamma_2, \dots, \gamma_k, \dots\}$.

При этом должен быть отмечен тот факт, что символ γ_k выступает на месте с номером k, т. е. мы должны кодировать и символы, и места.

. Мы будем кодировать символы, взаимно однозначно отображая их на множество нечетных чисел, бóльших единицы, по следующей схеме:

символы	γ_i:	*Л*	*П*	S_0	q_1	S_1	q_2	S_2	q_3 ...
числа	a_i:	3	5	7	9	11	13	15	17 ....

.. Мы будем кодировать имена мест, взаимно однозначно отображая их на множество простых чисел, расположенных в порядке возрастания:

номера мест:	1	2	3	4	5 ...	k ...
простые числа:	2	3	5	7	11 ...	p_k ....

... Выражению

$$`\gamma_{i_1}\gamma_{i_2}\dots\gamma_{i_n}'$$

отвечает по определению произведение

$$2^{a_{i_1}} \times 3^{a_{i_2}} \times \dots \times p_k^{a_{i_k}} \times \dots \times p_n^{a_{i_n}},$$

т. е. некоторое положительное целое число, которое мы обозначим через $\mathrm{ng}(M)$; это и есть гёделевский номер выражения M. Полу-

ченный гёделевский номер $\mathrm{ng}(M)$ допускает единственное разложение на множители, и ему отвечает единственное выражение M.

Чтобы код был полным, условимся кодировать пустое выражение целым числом 1.

Последовательности выражений. Аналогичным образом последовательность выражений

$$M_1,\ M_2,\ \ldots,\ M_j,\ \ldots,\ M_q$$

кодируется посредством числа

$$2^{\mathrm{ng}(M_1)} \times 3^{\mathrm{ng}(M_2)} \times \ldots \times p_j^{\mathrm{ng}(M_j)} \times \ldots \times p_q^{\mathrm{ng}(M_q)}.$$

а) Закодировать команду $(q_1 \mid \Pi\ q_2)$.

б) Декодировать выражение $2^{17}3^{11}5^{17}7^{11}$.

в) Закодировать последовательность $\varphi = (q_1 \mid \Pi q_2,\ q_3 \mid Bq_1)$.

г) Доказать, что никакое число не может быть одновременно гёделевским номером выражения и последовательности выражений. Доказать, что $\mathrm{ng}(\varphi) = \mathrm{ng}(\psi)$ влечет за собой $\varphi = \psi$. Для этого $\mathrm{ng}(\varphi)$ должно быть представлено в виде $2^m \cdot (2q + 1)$; затем следует рассмотреть все возможные случаи в зависимости от четности m.

§ 5.4. РЕКУРСИВНЫЕ И РЕКУРСИВНО ПЕРЕЧИСЛИМЫЕ МНОЖЕСТВА

5.4.0. Функции $Q_z^p(\mathfrak{X})$

Для любого целого числа z возможны два взаимоисключающих случая:

. либо не существует никакой машины Тьюринга, имеющей z в качестве своего гёделевского номера;

.. либо такая машина существует и притом только одна.

Мы будем говорить, что функция $f(\mathfrak{X})$,

$$\mathfrak{X} \in N^p \xrightarrow{f} f(\mathfrak{X}) \in N$$

частично вычислима, если для некоторой машины Тьюринга она является той функцией, которую эта машина вычисляет. Иначе говоря, всякой частично вычислимой функции $f(\mathfrak{X})$ можно сопоставить гёделевский номер z той машины Тьюринга, которая эту функцию вычисляет (может существовать несколько разных машин, вычисляющих одну и ту же функцию).

Теперь мы определим функции $Q_z^p(\mathfrak{X})$ следующим образом.

Пусть имеется пара, образованная целым неотрицательным числом z и входным заданием $\mathfrak{X} \in N^p$. Тогда:

. Если z есть гёделевский номер машины Z, вычисляющей частично вычислимую функцию $f(\mathfrak{X})$, то $Q_z^p(\mathfrak{X})$ совпадает с $f(\mathfrak{X})$.

.. Если z не является гёделевским номером никакой машины, то $Q_z^p(\mathfrak{x})$ принимает значение 0.

Ясно, что последовательность

$$Q_0^p(\mathfrak{x}),\ Q_1^p(\mathfrak{x}),\ \ldots,\ Q_n^p(\mathfrak{x}),\ \ldots$$

перечисляет, быть может с повторениями, множество частично вычислимых функций от p аргументов. Заметим (это понадобится нам в дальнейшем), что мы умеем, таким образом, перечислять области определения частично вычислимых функций.

5.4.1. Природа функций $Q_z^p(\mathfrak{x})$

Очевидно, что $Q_z^p(\mathfrak{x})$ есть функция от $p+1$ аргументов с областью определения N^{p+1}, принимающая значения из N.

Пусть заданы z и $\mathfrak{x}$. Тогда, чтобы вычислить значение Q_z^p, необходимо сначала выяснить, является ли z гёделевским номером какой-либо машины Тьюринга. Этот вопрос сводится к проблеме декодирования с ответами «да»/«нет», которую мы поручали машине Тьюринга G_2 (см. п. 5.3.2).

Если ответ «нет», то Q_z^p принимает значение 0.

Если ответ «да», то машине Z, команды которой описываются «анализом» ее номера z, предлагается $\mathfrak{x}$ в качестве входного задания.

Соединяя G_2 и Z, мы можем получить машину Тьюринга, вычисляющую $Q_z^p(\mathfrak{x})$. Тем самым доказана

Теорема. *Функции $Q_z^p(\mathfrak{x})$ частично вычислимы.*

Машина Тьюринга, вычисляющая функцию $Q_z^p(\mathfrak{x})$, называется *универсальной*: если поместить на ее ленте подходящее число z, то она сможет вычислять частично вычислимую функцию, соответствующую этому числу.

5.4.2. Замечание, относящееся к терминологии

Ниже, в п. 5.5.3, будут определены *рекурсивные функции*. Здесь же мы хотим только сделать следующее замечание.

Можно доказать, что класс вычислимых функций совпадает с классом рекурсивных функций; следовательно, прилагательные «вычислимый» и «рекурсивный» являются в принципе синонимами. Существуют, однако, случаи, в которых принято говорить лишь о рекурсивных функциях, избегая термина «вычислимые функции». Мы будем следовать именно этой традиции.

5.4.3. Определения

Далее нам часто придется пользоваться понятием характеристической функции множества.

Пусть $E \subset N$. Тогда *характеристическая функция* множества E, обозначаемая через C_E, определяется следующим образом:

$$C_E(x) = \begin{cases} 1, & \text{если } x \in E, \\ 0, & \text{если } x \notin E. \end{cases}$$

Аналогично определяется характеристическая функция множества $E \subset N^p$.

Рекурсивные множества. Мы будем говорить, что некоторое множество *рекурсивно*, если его характеристическая функция вычислима (ср. замечание в п. 5.4.2). Если множество рекурсивно, то существует машина Тьюринга, которая, получив на вход элемент этого множества, всегда ставит ему в соответствие некоторый ответ: либо 1, либо 0.

Рекурсивно перечислимые множества. Множество называется *рекурсивно перечислимым*, если оно является областью определения некоторой частично вычислимой функции.

Теоретически мы умеем перечислять частично вычислимые функции от одной целочисленной переменной, от двух целочисленных переменных и т. д.; следовательно, в принципе мы умеем перечислять и рекурсивно перечислимые множества: $\mathfrak{N}_0, \mathfrak{N}_1, \ldots, \mathfrak{N}_i, \ldots$.

5.4.4. Соотношение между понятиями рекурсивного и рекурсивно перечислимого множества

Теорема. *Множество является рекурсивным тогда и только тогда, когда оно само и его дополнение рекурсивно перечислимы.*

. Предположим, что множество R рекурсивно. Тогда существует машина Тьюринга, вычисляющая его характеристическую функцию (которая вычислима в силу рекурсивности R).

Отправляясь от этой машины, можно построить две другие машины Тьюринга, вычисляющие функции

$$f(\mathfrak{X}) = \begin{cases} 1, & \text{если } \mathfrak{X} \in R, \\ \text{не определена}, & \text{если } \mathfrak{X} \notin R; \end{cases}$$

$$\overline{f}(\mathfrak{X}) = \begin{cases} 0, & \text{если } \mathfrak{X} \in \overline{R}, \\ \text{не определена}, & \text{если } \mathfrak{X} \notin \overline{R}. \end{cases}$$

Отсюда следует, что R и $\overline{R}$ суть рекурсивно перечислимые множества.

.. Предположим, что множества R и $\overline{R}$ рекурсивно перечислимы. Тогда они являются областями определения частично вычислимых функций $f(\mathfrak{X})$ и $\overline{f}(\mathfrak{X})$ соответственно; этим функциям соответствуют вычисляющие их машины Тьюринга Z и $\overline{Z}$.

Пусть дано $\mathfrak{X}$. Если предложить его в качестве исходного задания машинам Z и $\overline{Z}$, то одна из них обязательно выдаст некоторый

результат, а другая нет. Если результат дает машина Z, то $C_R(\mathfrak{X}) = 1$, а если $\overline{Z}$, то $C_R(\mathfrak{X}) = 0$.

Пара Z, $\overline{Z}$ эквивалентна некоторой единой машине Тьюринга. Следовательно, функция C_R вычислима и R, равно как и $\overline{R}$, является рекурсивным множеством.

5.4.5. Неэквивалентность рекурсивности и рекурсивной перечислимости

Приведенная теорема оставляет открытым вопрос о том, существуют ли рекурсивно перечислимые, но *не рекурсивные* множества.

Мы докажем существование таких множеств с помощью так называемого «диагонального процесса».

Рассмотрим множество машин Тьюринга, вычисляющих функции от одного аргумента. Каждой из этих машин мы будем предлагать в качестве исходного задания ее собственный гёделевский номер. Возможно одно из двух: либо, начав вычисления с $z = \mathrm{ng}(Z)$, машина Z когда-нибудь остановится и выдаст некоторый результат (в таком случае машину называют *самоприменимой*), либо она никогда не остановится.

Таким способом мы получаем некоторую функцию от z, которая, очевидно, есть не что иное, как $Q^1_z(z)$; ясно, что это частично вычислимая функция. Можно построить машину Тьюринга, вычисляющую эту функцию; для этого надо сначала построить универсальную машину, вычисляющую $Q^1_z(x)$.

Таким образом, множество G гёделевских номеров самоприменимых машин оказывается рекурсивно перечислимым.

Теперь предположим, что оно является и рекурсивным. Это равносильно рекурсивной перечислимости его дополнения $\overline{G}$.

В таком случае должна существовать машина Тьюринга с каким-то номером λ, перечисляющая $\overline{G}$ (т. е. вычисляющая некоторую функцию с областью определения $\overline{G}$); в перечислении рекурсивно перечислимых множеств множество $\overline{G}$ носило бы именно этот номер:

$$\overline{G} = \mathfrak{N}_\lambda.$$

Для любого натурального числа x мы **имели бы следующее:**

$$x \in \overline{G} \Leftrightarrow x \in \mathfrak{N}_\lambda.$$

Однако в силу самого определения

$$x \in \overline{G} \Leftrightarrow x \notin \mathfrak{N}_x.$$

Следовательно,

$$x \in \mathfrak{N}_\lambda \Leftrightarrow x \notin \mathfrak{N}_x.$$

Если теперь положить $x = \lambda$, то

$$\lambda \in \mathfrak{N}_\lambda \Leftrightarrow \lambda \notin \mathfrak{N}_\lambda.$$

Полученное противоречие доказывает, что множество G не является рекурсивным.

5.4.6. Проблема остановки для произвольной машины Тьюринга

Когда некоторой машине Тьюринга предлагается исходное задание, возможны ровно два взаимоисключающих случая:

— машина когда-нибудь останавливается, выдавая определенный результат;

— машина работает вечно.

Можно сформулировать следующую проблему: существует ли машина Тьюринга T, для которой входными заданиями являются пары и которая, получив на вход пару

$$[z = \mathrm{ng}(Z);\ X],$$

отвечает

— либо «Да; машина Z, имеющая гёделевский номер z, начав вычисления с X, выдаст некоторый результат»;

— либо «Нет; предлагать машине Z задание X не следует — она никогда не остановится»?

Предположим, что такая машина T существует.

Пусть E — рекурсивно перечислимое, но не рекурсивное множество; *мы знаем, что такие множества существуют*!

Множество E есть область определения некоторой частично вычислимой функции f_{z_α}, вычисляемой машиной Z_α. Машина T для любого натурального числа x давала бы один из двух ответов:

«Да, $x \in E$ и Z_α остановится»;

«Нет, $x \notin E$ и Z_α не остановится».

Это означало бы, что E — рекурсивное множество, а такой вывод противоречит исходному предположению.

Следовательно, имеет место

Теорема. *Общая проблема остановки для произвольных машин Тьюринга неразрешима.*

§ 5.5. ЗАМЕЧАНИЯ И ДОПОЛНЕНИЯ

5.5.1. Реальные вычислительные машины

Мы называем машину U *универсальной*, если она вычисляет функцию $Q_z^0(\mathfrak{X})$. Когда этой машине в качестве исходного задания предложена пара $(z, \mathfrak{X})$, она выдаст тот результат, который получила бы, исходя из $\mathfrak{X}$, машина Z, гёделевский номер которой $\mathrm{ng}(Z) = z$.

Это замечание делает очевидной аналогию между универсальной машиной Тьюринга U и электронными вычислительными

машинами (ЭВМ). Машина U соответствует ЭВМ, а z — конкретной программе, которую вводят в ЭВМ вместе с исходным заданием и которая осуществляет над этим заданием вычислительные операции.

Тем самым проблема остановки (иначе, проблема применимости) обретает следующий конкретный смысл. Пусть имеется программа P и входные данные δ; спрашивается, существует ли такая общая программа G, которая позволяет заранее выяснить, остановится ли машина, начав вычисления по программе P с входными данными δ. Ответ является отрицательным: подобной программы G не существует. В частности, это означает, что никогда не будет написана программа, с помощью которой можно было бы обнаруживать в любых программах ошибки, приводящие к бесконечным вычислениям.

5.5.2. *«Философское» отступление*

Результаты, относящиеся к неразрешимости тех или иных проблем, отнюдь не дают повода для каких-либо философских рассуждений на тему о сравнении возможностей человека и машины Дело обстоит просто: в рамках некоторой математической теории удается доказать несуществование определенных объектов, точно так же как в арифметике доказывается несуществование рационального числа, квадрат которого был бы равен 2.

Открытие иррациональных чисел поставило перед древними греками целый ряд метафизических проблем; это объясняется тем, что подобное открытие опрокидывало «наивное» представление о мире, основанное на отношениях между целыми числами. Что же касается понятия неразрешимости, то оно ничего не опрокидывает. Неразрешимость той или иной проблемы означает всего лишь, что эта проблема плохо поставлена. Дальнейшее поведение исследователя определяется конкретной природой проблемы. Иногда целесообразно перейти к рассмотрению частных случаев, для которых данная проблема разрешима и может быть подвергнута математическому изучению. Бывает, однако, и так, что необходимо все же иметь дело с общим случаем (ибо именно он представляет прагматический интерес); тогда приходится «выкручиваться кто как может» — или, выражаясь более изящно, использовать *эвристические методы*[1]). Конкретный пример такой ситуации — поиск правил, позволяющих обнаружить ошибки программирования, ведущие к зацикливанию, весьма практичным, хотя и не на 100% надежным способом. Другой пример — поиск методов вычислений, для которых существовала бы общая процедура обнаружения ошибок.

---

[1]) Возможна также ситуация, когда решающий проблему алгоритм существует, но практически не может быть реализован ввиду своей чрезвычайной сложности. При этом нередко удается доказать, что для той или иной проблемы простых алгоритмов нет. — *Прим. ред.*

§ 5.3. Замечание о рекурсивных функциях

Для рекурсивных функций, играющих столь важную роль в математической логике, существует много различных (эквивалентных) определений. Следуя М. Дэвису, мы приведем здесь определение Джулии Робинсон. (Читатель сразу же поймет, как интерпретировать это определение с точки зрения теории машин Тьюринга.)

Определение. Функция называется *частично рекурсивной*, если ее можно построить путем конечного числа применений операций композиции и минимизации из следующих основных функций:

(1) $\mathrm{suc}(x) = x + 1$ (операция взятия следующего числа);

(2) $U_i^n(x_1, \ldots, x_n) = x_i$, $1 \leqslant i \leqslant n$ (нахождение i-й проекции вектора);

(3) $x + y$ (сложение);

(4) $x \dot{-} y$ («урезанное» вычитание);

(5) xy (умножение).

Частично рекурсивная функция называется *рекурсивной*, если определяющие ее операции разрешается применять лишь тогда, когда функции, получающиеся в результате их применения, регулярны [1]).

Основные результаты, относящиеся к рекурсивным функциям, таковы:

Теорема. *Если функция (частично) рекурсивна, то она (частично) вычислима.*

Теорема доказывается с помощью элементарных и стандартных машин Тьюринга. Доказательство ее несложно и предоставляется читателю в качестве упражнения.

Верна и обратная теорема.

Теорема. *Если функция (частично) вычислима, то она (частично) рекурсивна.*

Мы ограничимся тем, что наметим в самых общих чертах главную идею доказательства этой теоремы.

Прежде всего нам понадобится понятие предиката.

Определение. *n-местным предикатом* называется функция, определенная на некотором множестве n-ок и принимающая значения в множестве, состоящем из двух элементов {*истина*, *ложь*}.

Например, предикат $P(x, y) \equiv (x + y = 3)$ имеет значение *ложь* для пары $(1,1)$, значение *истина* для пары $(1, 2)$, значение *ложь* для пары $(4, y)$, каково бы ни было натуральное y, и т. д.

[1]) Достаточно было бы потребовать этого для операции минимизации, так как остальные операции сохраняют свойство регулярности функций. — *Прим. ред.*

Характеристическая функция n-местного предиката — это характеристическая функция множества n-ок, для которых данный предикат принимает значение *истина* (это множество называется *экстенсионалом* предиката).

Говорят, что предикат вычислим, если его характеристическая функция вычислима.

Предикат $P(x_1, \ldots, x_n)$ называется *полувычислимым*, если существует частично вычислимая функция $f(x_1, \ldots, x_n)$, определенная на его экстенсионале, т. е. на множестве, на котором P принимает значение *истина*; функция f не определена на дополнении этого множества, т. е. на множестве, на котором P принимает значение *ложь*.

Ясно, что всякий вычислимый предикат является и полувычислимым. Его характеристическая функция C (всюду определенная) принимает значение 0 (**NB**!) на его экстенсионале и значение 1 вне этого экстенсионала. Мы можем, например, определить соответствующую функцию f так:

$$f = \min_y [C + y = 0].$$

Поэтому имеет место

Теорема. *Предикат $R(x_1, \ldots, x_n)$ вычислим тогда и только тогда, когда предикаты R и $\neg R$ полувычислимы.*

Будем рассматривать предикаты, определенные на множестве цепочек, составленных из символов $\{S, q, z, и, Л, П, !\}$ (см. выше п. 5.3.1). Каждый такой предикат мы можем с помощью гёделевской нумерации превратить в некоторый числовой предикат. В частности, таким способом можно получить «числовые выражения» для различных предикатов, с помощью которых определяются машины Тьюринга и вычисления на этих машинах, например:

QUAD (x): истинен, если x — гёделевский номер команды машины Тьюринга; ложен в противном случае.

MT (x): истинен, если x — некоторый гёделевский номер некоторой машины Тьюринга (которая задана последовательностью команд); ложен в противном случае.

PASSE (x, y, z): истинен, если x и y — гёделевские номера конфигураций α и β и если в программе машины Тьюринга, имеющей гёделевский номер z, $\alpha \rightarrow \beta$; ложен в противном случае.

Теперь мы в состоянии наметить идею доказательства теоремы. Она состоит в следующем. а) Указываются некоторые общие способы получения одних «цепочечных» предикатов из других, такие, что при них соответствующие числовые предикаты сохраняют рекурсивность [т. е. если некоторый цепочечный предикат (или система предикатов) обладает тем свойством, что соответствующий

числовой предикат рекурсивен, то и для предиката, полученного из данного с помощью одного из этих способов, соответствующий числовой предикат будет рекурсивным]. б) Непосредственно доказывается, что некоторые «простейшие» цепочечные предикаты рекурсивны. в) Устанавливается, что для каждой машины Тьюринга T, вычисляющей функцию $f(x)$, из этих «простых» предикатов можно последовательным применением способов, указанных в п. а), получить предикат, для которого соответствующий числовой предикат $U(x, y, z)$ истинен тогда и только тогда, когда z есть гёделевский номер некоторого вычисления машины T, x — число палочек в начальной конфигурации этого вычисления, y — то же для заключительной конфигурации. Тогда функция f получится из характеристической функции предиката U с помощью операции минимизации.

5.5.4. Вычислимость и рекурсивность

В силу приведенных теорем вычислимость и рекурсивность эквивалентны. Понятие вычислимости ввел Тьюринг, а понятие рекурсивности — Клини. Оба эти понятия, определенные их авторами независимо друг от друга, были призваны формализовать интуитивное представление математиков о вычислимости. Примечательно, что все другие понятия, сконструированные с той же целью также независимо друг от друга и от обоих названных понятий (определения Гёделя, Чёрча, Поста, Маркова), оказались эквивалентными друг другу. Это позволило Чёрчу сформулировать свой знаменитый тезис (по существу философский) о том, что все упомянутые выше понятия правильно формализуют интуитивное представление о вычислимости и что это последнее невозможно обобщить. До сих пор опыт науки всегда подтверждал тезис Чёрча. С практической же точки зрения представляется более интересным ограничивать понятие вычислимости с тем, чтобы приспособить его к имеющимся в нашем распоряжении вычислительным средствам. В этом направлении ведутся многочисленные исследования[1]).

5.5.5. Упражнения

1. Доказать (во всех подробностях), что рекурсивность влечет вычислимость.

2. Показать, что следующие функции являются рекурсивными, указав способ, которым они могут быть получены из основных функций:

а) «нулевая» функция: $\forall x\ N(x) = 0$;

б) функция $\alpha(x)$, равная нулю для всех x, кроме $x = 0$, и единице для $x = 0$;

в) целая часть квадратного корня $\beta(x) = [\sqrt{x}]$;

---

[1]) Сюда прежде всего относится интенсивно развивающаяся в последние годы теория сложности вычислений (см. примечание на стр. 200). — *Прим. ред.*

г) целая часть частного: $\gamma(x, y)=\left[\frac{x}{y}\right]$, и остаток от деления x на y.

3. Доказать, что если R и S — рекурсивные множества, то множества $R \cup S$, $R \cap S$ и $\overline{R}$ также рекурсивны.

4. Доказать, что если P и Q — рекурсивные предикаты, то предикаты $P \vee Q$, $P \wedge Q$ и $\neg P$ также рекурсивны.

5. Доказать, что следующие предикаты вычислимы (использовать характеристические функции):

а) $x = y$;

б) $x < y$;

в) $x|y$ (y делится на x);

г) $\mathrm{Prem}(x)$ (x есть простое число).

Глава VI

КОМБИНАТОРНЫЕ СИСТЕМЫ И МАШИНЫ ТЬЮРИНГА: НЕРАЗРЕШИМЫЕ ПРОБЛЕМЫ

§ 6.1. КОМБИНАТОРНЫЕ СИСТЕМЫ И МАШИНЫ ТЬЮРИНГА

Мы описали два математических объекта, введенных с целью формализовать интуитивное понимание вычислимости, а именно:

машины Тьюринга;

комбинаторные системы.

Было указано, что машины Тьюринга эквивалентны рекурсивным функциям; теперь мы покажем, что они эквивалентны также и комбинаторным системам.

6.1.1. Тьюринговы вычисления и полутуэвские системы

Пусть Z — машина Тьюринга, команды которой распределяются по трем типам:

(1) $q_i\ S_j\ S_k\ q_l$,

(2) $q_i\ S_j\ П\ q_l$,

(3) $q_i\ S_j\ Л\ q_l$.

Напомним, что S_1 — это палочка «|» и что положительное целое число m изображается посредством $m+1$ палочек (иначе говоря, $m = |^{m+1}$).

Если предложить машине Z представление $\bar{m}$ числа m в качестве входного задания, то она начнет вычисление с конфигурации

$$q_1\bar{m}.$$

Это вычисление либо приведет к определенному результату, либо не даст никакого результата. Обозначим через P_Z множество целых неотрицательных чисел, для которых вычисления приводят к некоторым результатам; в дальнейшем мы будем интересоваться не самими этими результатами, а только их существованием.

Будем предполагать (мы имеем на это право в силу п. 5.1.1), что Z заключает все промежуточные результаты вычислений между двумя маркерами h.

Паре (Z, m) мы поставим в соответствие полутуэвскую систему $(ST)_1$ с аксиомой $hq_1\bar{m}h$; продукции этой системы выберем таким образом, чтобы некоторые ее теоремы соответствовали конфигурациям машины Z, в то время как другие теоремы позволили бы нам сопоставить заключительной конфигурации теорему

$$hrh,$$

означающую: «Z применима к m, т. е. соответствующее вычисление приводит к некоторому результату».

Проведем формальное построение системы $(ST)_1$.

(а) Каждой команде типа (1) $q_iS_jS_kq_l$ машины Z сопоставляется следующая продукция:

$$Pq_iS_jQ \to Pq_lS_kQ.$$

Эта продукция обеспечивает переход от конфигурации Pq_iS_jQ к той конфигурации, которая получается из нее применением данной команды.

(б) Каждой команде типа (2) $q_iS_jПq_l$ сопоставляются все продукции вида

$$PS_kq_iS_jQ \to Pq_lS_kS_jQ$$

(этих продукций столько, сколько имеется разных S_k). Их задача также состоит в переходе от конфигурации $PS_kq_iS_jQ$ к той конфигурации, которую дает применение рассматриваемой команды.

(в) Каждой команде типа (3) $q_iS_jЛq_l$ ставится в соответствие продукция

$$Pq_lS_jQ \to PS_jq_lQ,$$

которая выполняет ту же задачу, что и продукции типов (а) и (б).

Описанные до сих пор продукции соответствуют тем ситуациям машины Z, в которых применимы ее команды. Рассмотрим теперь те ситуации, в которых Z останавливается, т. е. ситуации, выступающие только в заключительных конфигурациях. Мы собираемся заменять результаты вычислений единственным ответом hrh (существование результата). Чтобы сделать это, введем продукции другого рода, применимые только к тем теоремам системы $(ST)_1$, которые соответствуют заключительным конфигурациям. Эти продукции таковы:

(г) $$Pq_iS_jQ \to PqS_jQ$$

для всех пар (q_i, S_j), не являющихся левыми частями команд машины Z; эти продукции вводят маркер q;

(д) $$PqS_jQ \to PqQ$$

— продукции, стирающие все символы, отличные от h, справа от маркера q;

(е) $$PqhQ \to PrhQ$$

— продукция, вводящая маркер r слева от маркера h;

(ж) $$PS_jrQ \to PrQ$$

— продукции, стирающие слева от r все символы, за исключением h.

Теперь мы можем сформулировать и доказать основную теорему данного раздела.

Теорема. *Машина Тьюринга Z применима к m в том и только в том случае, если выражение hrh является теоремой полутуэвской системы $(ST)_1$ с аксиомой $hq_1\overline{m}h$, поставленной в соответствие машине Z посредством описанного выше построения.*

Используя символ Фреге «$\vdash$» (его значение будет понятно из контекста), мы можем записать эту теорему (вернее, метатеорему) следующим образом:

$$m \in P_Z \Leftrightarrow \underset{(ST)_1}{\vdash} hrh.$$

1) Доказательство «слева направо». Включение $m \in P_Z$ означает, что существует вычисление, начинающееся с m при состоянии q_1. Иначе говоря, существует конечная последовательность конфигураций

$$\alpha_1, \alpha_2, \ldots, \alpha_n,$$

где $\alpha_1 = hq_1mh$, α_n — заключительная конфигурация и $\alpha_i \to \alpha_{i+1}$, $1 \leqslant i < n$.

Таким образом, мы имеем $\underset{(ST)_1}{\vdash} \alpha_n$, и ситуация, фигурирующая в α_n, не встречается в левых частях команд машины Z. Но к α_n можно применить продукцию типа (г) $Pq_iS_jQ \to PqS_jQ$, а затем — продукции типов (д), (е) и (ж), так что в конце концов получится hrh.

2) Доказательство «справа налево». Из рассмотрения продукций типов (г), (д), (е) и (ж) вытекает, что любое доказательство, ведущее к hrh, обязательно проходит через теорему вида hP_0qQ_0h и что в $(ST)_1$ для hP_0qQ_0h существует доказательство $M_1, M_2, \ldots, M_n$.

Таким образом, мы имеем $M_1 = hq_1\overline{m}h$, $M_n = hP_0qQ_0h$ и $M_i \to M_{i+1}$, $1 \leqslant i < n$.

Существует целое число s, такое, что $s < n$ и что всякое M_i, $1 \leqslant i \leqslant s$, содержит некоторое q_{α_i}, причем M_s есть заключительная конфигурация машины Z. Следовательно, $m \in P_Z$.

Замечания о строении системы $(ST)_1$. Следует обратить внимание на то, что в $(ST)_1$ от числа m зависит только аксиома $hq_1\overline{m}h$; алфавит и продукции системы от m не зависят. В качестве упражнения можно доказать, что система $(ST)_1$ моногенна (см. ниже п. 6.1.5, упр. 1).

Если m принимает значения из множества неотрицательных целых чисел, то мы получим семейство полутуэвских систем типа $(ST)_1$. С помощью этого семейства множеству тех чисел m, к которым применима Z, ставится в соответствие *единственная* теорема hrh.

Эта ситуация подсказывает мысль об обращении указанного подхода с тем, чтобы построить систему, которая, исходя из *аксиомы hrh*, выдавала бы в качестве заключительных теорем все целые числа, к которым применима Z.

6.1.2. Машины Тьюринга и полутуэвские системы

Рассмотрим систему $(ST)_2$, полученную из семейства систем типа $(ST)_1$ следующим образом:

. аксиомой системы $(ST)_2$ является выражение hrh;

.. продукциями системы $(ST)_2$ являются обращения продукций систем $(ST)_1$ (напомним, что все эти системы имеют одно и то же множество продукций).

Проведенное выше исследование систем типа $(ST)_1$ показывает, что следующие утверждения равносильны: «$hq_1\bar{m}h$ является начальной конфигурацией машины Z, приводящей к некоторому результату» и «$hq_1\bar{m}h$ является теоремой системы $(ST)_2$».

Теорема. *Машина Тьюринга Z применима к m в том и только в том случае, если выражение $hq_1\bar{m}h$ является теоремой полутуэвской системы $(ST)_2$, которая получается «обращением» систем $(ST)_1$, сопоставляемых парам (Z, m).*

Замечание. Можно дополнить $(ST)_2$ продукциями, позволяющими стирать h и q_1. Построенная таким образом система $(ST)_3$ позволяет записать наш результат более элегантно:

$$m \in P_Z \Leftrightarrow \underset{(ST)_3}{\vdash} m.$$

6.1.3. Машины Тьюринга и туэвские системы

В предыдущем пункте машине Тьюринга Z ставилась в соответствие полутуэвская система; теперь мы поставим ей в соответствие туэвскую систему.

Рассмотрим систему (T) с тем же алфавитом, что и раньше, имеющую

. аксиому hrh;

.. множество продукций, представляющее собой объединение множеств продукций систем $(ST)_1$ и $(ST)_2$.

Система (T) является туэвской, поскольку продукции системы $(ST)_2$ суть обращения продукций систем $(ST)_1$.

Покажем, что множество теорем системы (T) совпадает с множеством теорем системы $(ST)_2$.

В силу нашего построения очевидно, что всякая теорема системы $(ST)_2$ есть теорема системы (T): в самом деле, (T) имеет ту же аксиому, что и $(ST)_2$, и располагает всеми продукциями системы $(ST)_2$.

Перейдем к обратному утверждению.

Обозначим через R_1, R_2, ..., R_t продукции системы $(ST)_2$ и через S_1, S_2, ..., S_t — их обращения, т. е. продукции системы $(ST)_1$.

Нетрудно заметить, что введение S_i позволяет «удлинять» доказательство; например, «кусок» доказательства

$$M \text{ (в силу } R_1) \to N \text{ (в силу } R_2) \to P$$

можно «удлинить» следующим образом:

$$M \text{ (в силу } R_1) \to N \text{ (в силу } S_1) \to M \text{ и т. д.}$$

По существу нам нужно доказать, что (T) не содержит никаких теорем, помимо теорем системы $(ST)_2$, хотя и может содержать более длинные доказательства.

Итак, пусть имеется доказательство слова M в (T). Выкинем все «бесполезные» шаги, полученные повторениями, и доказательство примет вид

$$hrh = M_1, M_2, \ldots, M_p = M.$$

Возможно, что в этом доказательстве выступают только продукции системы $(ST)_2$; в таком случае теорема доказана.

Допустим поэтому, что при переходе от M_j к M_{j+1} $(1 \leqslant j < p)$ появляется в *первый* раз продукция типа S, т. е. продукция, отсутствующая в $(ST)_2$.

Может ли M быть аксиомой hrh? Мы видели, что в $(ST)_1$ выражение hrh является заключительной теоремой, так что ни одна обратная продукция, т. е. продукция из $(ST)_2$, не может привести к hrh. Поэтому M_j должно быть следствием из M_{j-1} в силу некоторой продукции R_i; другими словами, M_{j-1} выводится из M_j с помощью применения правила типа S.

Однако тогда получается, что из M_j с помощью продукций типа S можно получить следствия M_{j+1} и M_{j-1}, которые должны быть различными в силу того факта, что из доказательства слова M были устранены все повторения. Этот результат противоречит моногенности системы $(ST)_1$.

Итак, доказана

Теорема. *Множество теорем туэвской системы* (T) *совпадает с множеством теорем полутуэвской системы* $(ST)_2$.

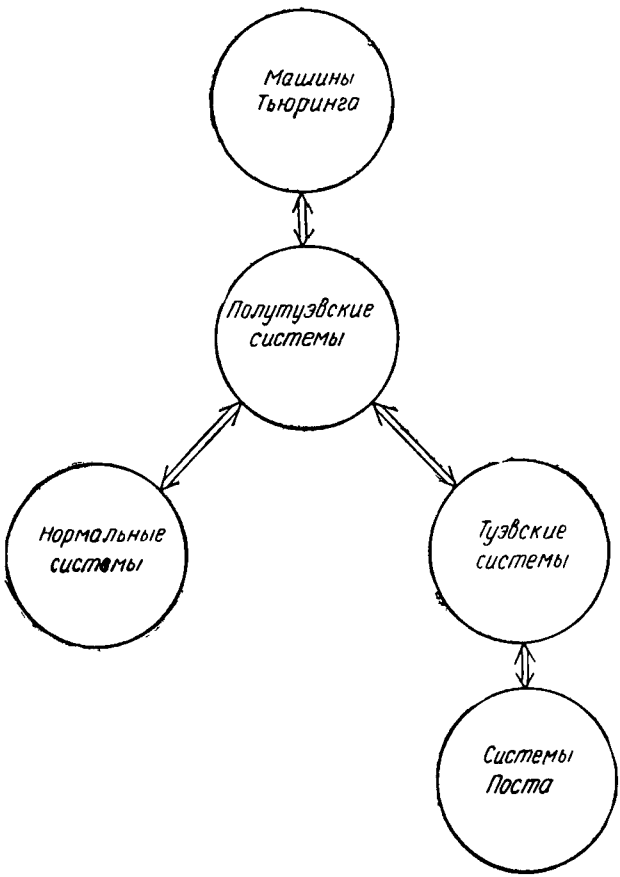
Иначе говоря, продукции, добавляемые к $(ST)_2$, чтобы получить (T), ведут не к новым теоремам, а только к некоторым вариантам доказательств.

6.1.4. Резюме полученных результатов

На протяжении нескольких последних глав мы доказали ряд теорем об эквивалентности различных концепций алгоритма. Добавим к ним еще следующий результат (его доказательство

предоставляется читателю в качестве упражнения): *всякой туэвской системе можно поставить в соответствие систему Поста с тем же множеством теорем.*

Общую картину можно представить следующий схемой:



Всякое множество, порождаемое одной из указанных на схеме систем, может быть порождено и любой другой из них; для любой данной системы может быть эффективно построена эквивалентная ей система другого типа.

Из эквивалентности машин Тьюринга и формальных систем следует

Теорема. *Всякое рекурсивно перечислимое множество может быть порождено любой из следующих систем: полутуэвской системой, туэвской системой, нормальной системой, системой Поста.*

6.1.5. Упражнения

1. Доказать, что $(ST)_1$ — моногенная система.
2. Доказать, что в туэвской системе для всякой теоремы существует доказательство без повторений.

§ 6.2. НЕРАЗРЕШИМЫЕ ПРОБЛЕМЫ

6.2.1. Следствия из неразрешимости проблемы остановки

Выше мы рассматривали следующую проблему: существует ли общий алгоритм (машина Тьюринга), который всегда отвечал бы однозначно («да» или «нет») на любой вопрос следующего типа: «Если данная машина Z начнет вычисления с данного числа m, то получит ли она результат?»

В общем виде эта проблема решается отрицательно. (Внимание: в *некоторых частных случаях* тот же самый вопрос может получить положительный ответ!)

Построение систем $(ST)_2$ и (T), соответствующих машине Z в смысле § 6.2, позволяет сопоставить аргументам машины Z теоремы систем $(ST)_2$ и (T). Выбрав в качестве Z такую машину, для которой проблема остановки неразрешима, мы увидим, что в соответствующих туэвской и полутуэвской системах рекурсивно неразрешима проблема доказуемости. (Эта проблема состоит в следующем: можно ли указать алгоритм, позволяющий для любого слова в алфавите системы отвечать «да» или «нет» на вопрос о том, является ли это слово теоремой системы?)

Строя нормальную систему и систему Поста, соответствующие полутуэвской и туэвской системам (см. § 3.2), мы также получим системы, в каждой из которых проблема доказуемости неразрешима.

6.2.2. Проблема тождества

Вернемся теперь к проблеме тождества, или эквивалентности слов, сформулированной в первой главе. Пусть некоторая факторполугруппа свободной полугруппы задается с помощью n соотношений Туэ: $P_1 \sim Q_1, \ldots, P_n \sim Q_n$; спрашивается, может ли быть установлено алгоритмически, эквивалентны ли в этой факторполугруппе два заданных слова A и B?

Рассмотрим туэвскую систему с аксиомой A и продукциями $P_1 \leftrightarrow Q_1, \ldots, P_n \leftrightarrow Q_n$.

Установить, эквивалентно ли слово A слову B, или выяснить, является ли B теоремой данной туэвской системы, — это равносильные проблемы.

Возьмем теперь построенную нами туэвскую систему с неразрешимой проблемой доказуемости и сопоставим ей полугруппу, полученную путем замены каждой пары туэвских продукций соответствующим соотношением Туэ; ясно, что для этой полугруппы проблема тождества неразрешима.

6.2.3. Проблема соответствий Поста

Пусть имеются следующие русские слова[1]:
M_1 = «ГА» (сокращение от *гектар*),
N_1 = «А» (союз),
M_2 = «ПИР» (праздник с обильной едой),
N_2 = «ПИ» (буква п),
M_3 = «О» (предлог),
N_3 = «РОГ» (у коровы).

Рассмотрим слово

«ПИРОГА» (полинезийская лодка).

Это слово может быть разбито на части двумя способами

ПИР/О/ГА

и

ПИ/РОГ/А;

тем самым из M можно сделать две шарады:

$$M = M_2M_3M_1 = N_2N_3N_1,$$

исходя из троек (M_1, M_2, M_3) и (N_1, N_2, N_3) и соблюдая следующее условие: если слово M_i выступает в одной шараде на j-м месте, то соответствующее слово N_i выступает в другой шараде также на j-м месте.

Проблема двойной шарады. Пусть имеются две n-ки слов в алфавите $\mathfrak{A}$

$$(G_1, \ldots, G_n) \quad \text{и} \quad (H_1, \ldots, H_n);$$

спрашивается, существует ли конечная последовательность индексов $i_1, i_2, \ldots, i_k$, такая, что можно построить двойную шараду

$$G_{i_1}G_{i_2} \ldots G_{i_k} = H_{i_1}H_{i_2} \ldots H_{i_k} \qquad (i_j \in \{1, 2, \ldots, n\}).$$

На этот *частный* вопрос иногда можно ответить положительно (построив, быть может, соответствующий пример), а иногда — отрицательно (указав какую-либо принципиальную невозможность например, построить двойную шараду невозможно, если всякое G_i длиннее, чем соответствующее H_i).

Проблема Поста. Здесь, однако, встает и другая, *общая* проблема, сформулированная Постом: «Существует ли общий алгоритм, который для всяких двух n-ок слов в данном алфавите решал бы, существует ли последовательность индексов, обеспечивающая указанное соответствие?»

---

[1]) В оригинале французский пример: **regorgeais** = re/gorge/ais = reg/or/geais. — *Прим. перев.*

Наметим ход рассуждения, с помощью которого доказывается неразрешимость этой проблемы.

Эскиз доказательства. Пусть имеются две конкретные n-ки в алфавите $\mathfrak{A}$:

$$(G_1, \ldots, G_n) \quad \text{и} \quad (H_1, \ldots, H_n).$$

Построим нормальную систему (N), в которой ни одно выводимое из аксиомы слово не пусто.

. Аксиома системы (N) — непустое слово $A \in \mathfrak{A}^*$.

.. Схемы продукций системы (N) таковы:

$$G_iP \to PH_i, \qquad 1 \leqslant i \leqslant n.$$

Исследуем процесс получения теорем в (N). Предположим, что аксиому A можно представить в виде

$$A = G'P',$$

где G' — одно из G_i; тогда появится возможность вывести из A следствие:

$$A = G'P' \to P'H',$$

где через H' обозначено H_i, соответствующее G_i.

Предположим, далее, что $P'H'$ представляется, в свою очередь, в виде

$$P'H' = G''P'',$$

где через G'' также обозначено одно из G_i; тогда мы получим, что

$$G''P'' \to P''H'' \text{ и т. д.}$$

Таким образом, доказательство в (N) производится по следующей схеме (отношение следования обозначается вертикальной стрелкой):

$$\begin{array}{ll}
(0) & A = G'P' \\
 & \quad\downarrow \\
(1) & \quad P'H' = G''P'' \\
 & \qquad\downarrow \\
(2) & \qquad P''H'' = G'''P''' \\
\cdots & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
(k-1) & \qquad\qquad\qquad\qquad P^{(k-1)}H^{(k-1)} = G^{(k)}P^{(k)} \\
 & \qquad\qquad\qquad\qquad\qquad\downarrow \\
(k) & \qquad\qquad\qquad\qquad\qquad P^{(k)}H^{(k)} = B
\end{array}$$

Из этой схемы непосредственно получаем

$$(0)' \quad AH' = G'P'H' = G'G''P''$$

$$(1)' \quad AH'H'' = G'G''P''H'' = G'G''G'''P'''.$$

. . . . . .

$$(k-2)' \quad AH'H'' \ldots H^{(k-1)} = G'G'' \ldots G^{(k-1)}P^{(k-1)}H^{(k-1)} =$$
$$= G'G'' \ldots G^{(k-1)}G^{(k)}P^{(k)}$$

$$(k-1)' \quad AH'H'' \ldots H^{(k-1)}H^{(k)} = G'G'' \ldots G^{(k-1)}G^{(k)}P^{(k)}H^{(k)} =$$
$$= G'G'' \ldots G^{(k-1)}G^{(k)}B.$$

Поскольку все H_i и G_i не пусты (хотя P_i могут в принципе быть пустыми), верны следующие неравенства:
длина слова $AH' \geqslant$ длина слова $G'G''$;
длина слова $AH'H'' \geqslant$ длина слова $G'G''G'''$ и т. п.
Итак, мы получили

Предложение. *Если B — теорема системы (N), то существует последовательность индексов, ведущая к «псевдопостовскому» равенству*

$$AH'H'' \ldots H^{(k)} = G'G'' \ldots G^{(k)}B,$$

а также к неравенствам

длина слова $AH' \ldots H^{(j)} \geqslant$ длина слова $G' \ldots G^{(j)}$.

Обратно, если имеют место это равенство и соответствующие неравенства, то можно «восстановить» все P_i и доказательство теоремы B (провести нужное рассуждение предоставляется читателю в качестве упражнения).

Мы видим, таким образом, что проблема установления того, выводимо ли данное слово из аксиомы в системе (N), эквивалентна «псевдопостовской» проблеме. Чтобы перейти к проблеме Поста, необходимо освободиться от неравенств, а также от самих слов A и B.

Можно эффективно построить (см. упражнения) такую нормальную систему, производную от (N), что ее проблема тождества эквивалентна проблеме Поста, а это влечет за собой неразрешимость последней.

Проблема Поста в свою очередь будет служить отправной точкой для доказательства некоторых теорем о неразрешимости, относящихся к формальным грамматикам.

Одна возможная интерпретация проблемы Поста. Пусть $\mathfrak{A}$ и $\mathfrak{B}$ — два алфавита (возможно, совпадающие).

Рассмотрим такое отображение свободной полугруппы $\mathfrak{A}^*$ в (или на) полугруппу $\mathfrak{B}^*$, которое было бы совместимо с кон-

катенацией (это значит, что образ произведения, т. е. конкатенации, должен совпадать с произведением образов). Подобное отображение является гомоморфизмом полугруппы; оно определяется образами однобуквенных слов.

Если даны два таких гомоморфизма φ и ψ, то можно поставить следующий вопрос: существует ли слово $M \in \mathfrak{A}^*$, имеющее одинаковые образы относительно φ и ψ, т. е. такое, что

$$\varphi(M) = \psi(M).$$

Эта проблема известна как проблема диагонализации двух гомоморфизмов свободной полугруппы. Очевидно, что она эквивалентна проблеме Поста.

6.2.4. *Неразрешимые проблемы: резюме*

Мы построили математический объект (машину Тьюринга), с которым связана некоторая неразрешимая проблема, и с помощью теорем об эквивалентности получили ряд результатов о неразрешимости, связанных с другими объектами. Последовательность изложения может быть представлена диаграммой на стр. 110.

6.2.5. *Упражнения*

1. Привести конкретные примеры ситуаций, в которых проблема соответствий Поста разрешима; сделать это, либо указав последовательность индексов, дающую положительный ответ, либо доказав, что такой последовательности не существует.

2. Устранить «лишние» условия в «псевдопостовской» проблеме.

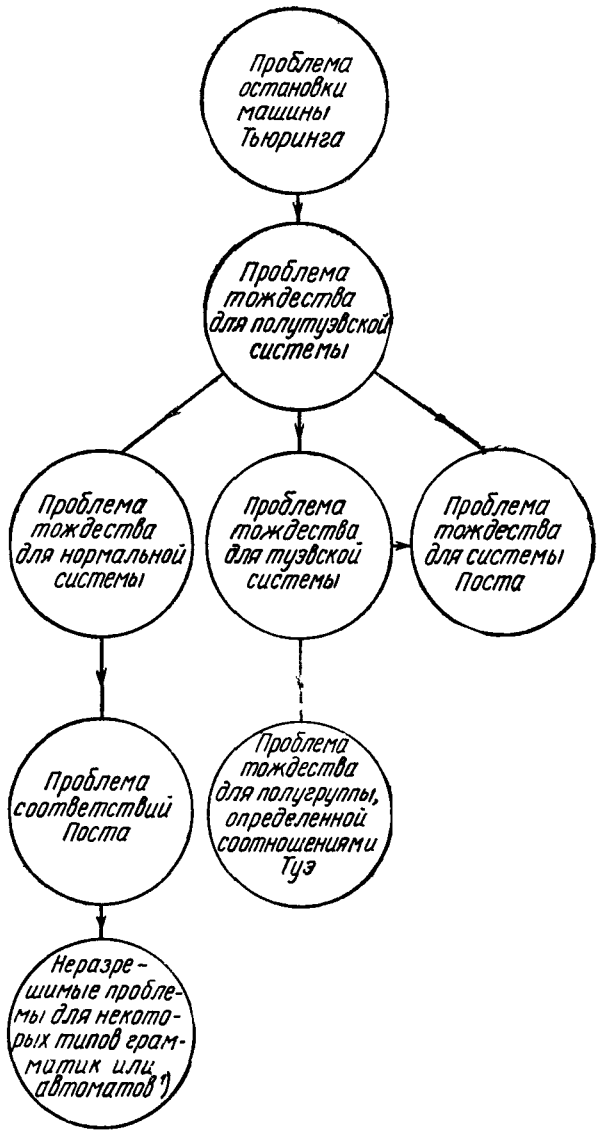
А. В свое время мы доказали, что любой системе можно сопоставить эквивалентную путем кодирования исходной системы в двухбуквенном алфавите $\{a, b\}$.

Предположим, что (N) заранее была подвергнута такому кодированию. Поставим в соответствие системе (N) «антинормальную» систему (AN), аксиома и левые (правые) части продукций которой представляют собой зеркальные образы соответственно аксиомы и левых (правых) частей продукций системы (N).

Затем дополним (AN) до системы $(AN1)$, алфавит которой содержит маркер h, причем этот маркер добавляется справа к каждому слову, выводимому в (AN).

Наконец, построим *нормальную* систему (NC), в которой выводимы те и только те слова, которые либо выводимы в системе $(AN1)$ либо получаются из слов, выводимых в $(AN1)$, циклическими перестановками:

$$(NC)\begin{cases} \text{Алфавит:} & \{a, b, h\}; \\ \text{Аксиома:} & \tilde{A}h; \\ \text{Продукции:} & \{\tilde{G}_i hP \to Ph\tilde{H}_i \mid 1 \leqslant i \leqslant n\}, \\ & aP \to Pa, \quad bP \to Pb, \quad hP \to Ph. \end{cases}$$



[1]) Рассматриваются ниже, в гл. VIII.

Продукции системы (NC) имеют вид

$$\{K_iP \to P\overline{K}_i \mid 1 \leqslant i \leqslant n+3\}.$$

Провести сравнение проблемы тождества для (NC) с псевдопостовской проблемой; показать, что неравенства здесь оказываются излишними. Использовать число вхождений маркера h.

Б. Теперь нам надо устранить слова A и B, вернее, $-\bar{A}h$ и $\bar{B}h$. Будем исходить из «псевдопостовского равенства». Закодируем все элементы таким образом, чтобы получить независимое равенство слов A и B (равенство Поста); можно показать, что оно осуществляется лишь одновременно с псевдопостовским равенством.

Введем $n+5$ пар слов $(F_i, \overline{F}_i)$:

если $K_i = x_1x_2 \ldots x_p$, то $F_i = x_1kx_2k \ldots x_pk$;

если $\overline{K}_i = y_1y_2 \ldots y_p$, то $\overline{F}_i = ky_1ky_2 \ldots ky_p$,

где $x_j, y_j \in \{a, b, h\}$ и k — некоторая новая буква. Последовательности $\bar{A}h = x_1x_2 \ldots x_j$ сопоставляется пара

$$(F_{n+4}, \overline{F}_{n+4}) = (kk, kkx_2 \ldots kx_j).$$

Последовательности $\bar{B}h = y_1y_2 \ldots y_j$ сопоставляется пара

$$(F_{n+5}, \overline{F}_{n+5}) = (y_1ky_2k \ldots y_jkk, kk).$$

Проделанные построения позволяют доказать наше предложение путем анализа возможных случаев. Следует воспользоваться методом доказательства от противного, рассматривая расположение вхождений k в интересующие нас слова.

Часть II
НЕКОТОРЫЕ ЗАМЕЧАТЕЛЬНЫЕ КЛАССЫ ЯЗЫКОВ

Глава VII
КОНТЕКСТНО-СВОБОДНЫЕ ЯЗЫКИ (ЯЗЫКИ ХОМСКОГО): ОБЩАЯ ХАРАКТЕРИСТИКА И ОСНОВНЫЕ СВОЙСТВА

§ 7.1. ГРАММАТИКА И ОПИСАНИЕ СИНТАКСИЧЕСКИХ СТРУКТУР

В начальных главах книги мы определяли язык как часть свободной полугруппы, порождаемой буквами некоторого алфавита (словами некоторого словаря) и грамматикой, выступающей как алгоритм, который позволяет перечислять слова (фразы) языка. Уточнения, которые мы внесли затем в представление об алгоритмах, дают нам возможность сформулировать ряд весьма общих результатов, относящихся к грамматикам и языкам.

7.1.1. Грамматики и машины Тьюринга

Итак, мы рассматриваем грамматику как алгоритм [1]), а понятие алгоритма отождествляем с понятием машины Тьюринга (или комбинаторной системы). С точностью до вариантов изложения мы можем теперь дать самое общее определение грамматики в терминах машин Тьюринга (или комбинаторных систем). Сказать, что грамматика G некоторого языка L, определенного над словарем V, есть перечисляющий алгоритм, — значит утверждать, что существует машина Тьюринга Z_G, которая сопоставляет числу 1 фразу φ_1, числу 2 — фразу φ_2, ..., любому натуральному числу n — фразу φ_n, так что производимое машиной Z_G рекурсивно перечислимое множество $\{\varphi_1, \varphi_2, \ldots, \varphi_n, \ldots\}$ совпадает с L.

Этот алгоритм (или другой алгоритм, родственный ему) может быть представлен как «средство распознавания». Именно, машине предлагается фраза φ, и если эта фраза принадлежит языку — и только в этом случае, — машина должна остановиться после конечного числа операций. Таким образом:

. если машина останавливается, φ принадлежит языку;

.. если же машина не остановилась, мы ничего не можем сказать о принадлежности фразы φ языку.

Однако в классе всех рекурсивно перечислимых языков не существует распознающего алгоритма, который всегда давал бы от-

---

[1]) Ср. примечание к стр. 66. — *Прим. ред.*

вет «да» или «нет» на вопрос о принадлежности любой фразы данному языку.

Классы грамматик. Крайняя общность только что приведенных определений не позволяет ухватить некоторые специфические особенности реально существующих языков (естественных и искусственных) и соответствующих грамматик. Поэтому данные понятия необходимо разумным образом сузить.

Налагая на грамматики дополнительные условия, мы можем получить иерархию классов грамматик, последовательно вложенных друг в друга. Разным классам грамматик отвечают более специфические (нежели машины Тьюринга) классы автоматов или комбинаторных систем.

Можно было бы определить все интересные классы грамматик сразу; однако, учитывая дидактические цели настоящей книги, мы предпочли начать с рассмотрения одного класса грамматик, занимающего в иерархии среднее положение. Этот класс естественным образом появляется при изучении синтаксиса; мы охарактеризуем его сперва эвристически.

7.1.2. Об одной весьма распространенной структуре

В качестве примеров мы будем брать французские фразы. Примем допущение, что мы всегда умеем установить, является ли данная последовательность французских слов фразой. При синтаксическом анализе будем опираться на понятия школьной грамматики.

Говоря о грамматике, мы будем пользоваться именами синтаксических категорий, такими, как «существительное», «глагол», «именная группа» и т. д. Ясно, что если слово *существительное* само является существительным, то слово *глагол* — не глагол; категория и обозначающее ее слово (или словосочетание) должны строго различаться.

Аналогичным образом в формальных грамматиках необходимо разграничивать терминальный, или основной, словарь (словарь символов собственно языка) и вспомогательный словарь, который используется только для формулирования правил.

Структуры фраз. Фраза, как правило, имеет в своем составе одно или несколько (однородных) подлежащих, одно или несколько сказуемых, одно или несколько дополнений, а также определения и обстоятельства.

Фразе

(1) *Riquet aboie* 'Рике лает'

отвечает очень простая схема

(I) существительное (nom) + глагол (verbe).

Подлежащее может иметь при себе различные определители; вместе с ними оно составляет именную группу, являясь ее «ядром». Так, фраза

(2) *Le chien famélique aboie* 'Голодная собака лает' имеет схему

(II) именная группа (groupe nominal) + глагол.

Посредством аналогичных расширений можно получать фразы вроде

(3) *Le chien famélique aboie plaintivement après la caravane* 'Голодная собака жалобно лает на караван'; эта фраза построена по схеме

(III) именная группа + глагольная группа + предлог + именная группа.

Введем следующие обозначения:

Ph — фраза;
Gn — именная группа;
Gv — глагольная группа;
Pr — предлог;
N — существительное;
Adj — прилагательное;
Art — артикль;
Adv — наречие;
V — глагол.

Теперь мы сможем записывать правила следующим способом, который будет пояснен на примерах. Например,

$$Ph \rightarrow \text{'}Gn\ Gv\ Pr\ Gn\text{'};$$

это означает: «Чтобы получить фразу, можно, в частности, взять именную группу, поместить справа от нее глагольную группу, затем предлог и, наконец, еще одну именную группу».

Другие примеры правил:

$$Gn \rightarrow \text{'}Art\ N\ Adj\text{'},$$

$$Gv \rightarrow \text{'}V\ Adv\text{'}.$$

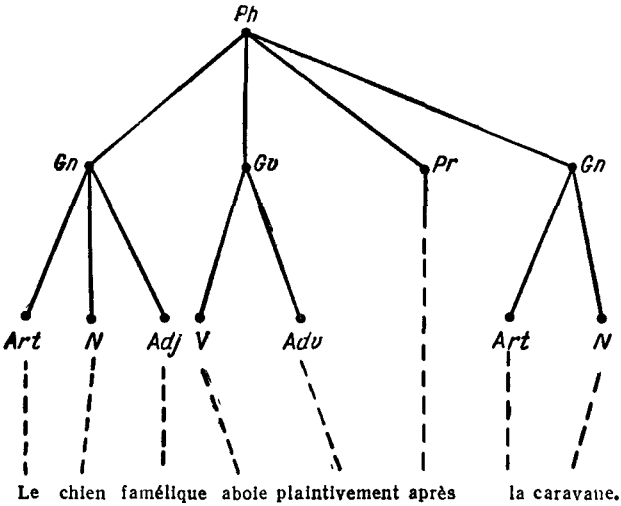
Дерево фразы. Удобный способ представлять синтаксическую структуру фраз состоит в использовании скобок, помеченных символами синтаксических категорий. Так, фраза (3) имеет следующую структуру:

$$\underset{Ph}{[}\ \underset{Gn}{[}\ \underset{Art}{[}\ Le]\ \underset{N}{[}\ chien]\ \underset{Adj}{[}\ famélique]\ ]\ \underset{Gv}{[}\ \underset{V}{[}\ aboie]\ \underset{Adv}{[}\ plaintivement]\ ]$$

$$\underset{Pr}{[}\ après]\ \underset{Gn}{[}\ \underset{Art}{[}\ la]\ \underset{N}{[}\ caravane]\ ]\ ].$$

Кроме того, синтаксическая структура фразы может быть изображена посредством дерева с помеченными узлами (пример см. ниже). Эти представления эквивалентны, и переход от одного

из них к другому выполняется чисто механически.



То, что для представления синтаксической структуры фраз выбраны именно деревья, является в некоторых отношениях актом произвола. В самом деле, можно было бы использовать графы более общего вида: например, с циклами. Данный выбор продиктован рядом эмпирических соображений. Деревья — достаточно простой тип графов; и хотя с их помощью невозможно выразить некоторые синтаксические явления (как, например, анафорические связи [1])), их использование дает, по крайней мере в первом приближении, удовлетворительные результаты. Иначе говоря, простота достигается не слишком дорогой ценой.

Для языков программирования положение остается в общих чертах тем же самым: программа представляет собой структуру,

---

[1]) Анафорическими связями обычно называют связи «отсылочного» типа, например связь между местоимением-заменителем *он, она, оно, они* и заменяемым им словом. Так, во фразе *Маша подвела дочку к дивану, затем она уложила ее на него* три анафорических связи: *она — Маша, ее — дочку* и *на него — на диван.* Другие примеры анафорических связей: связь между союзным словом типа *который, где, когда, ...* и той словоформой, к которой относятся вводимые этими словами придаточные; связь между указательным местоимением типа *этот, тот* и той именной группой, на которую оно указывает; и т. п.

Заметим, что в собственно синтаксической структуре (дереве) фразы анафорические связи и не должны представляться; по крайней мере они не должны представляться тем же самым образом, которым представляются в дереве «настоящие» (т. е. структурные, иерархические) синтаксические связи. — *Прим. перев.*

построенную из последовательностей, принадлежащих к разным синтаксическим категориям.

Грамматики и структуры фраз. Дерево фразы играет важную роль в описании синтаксиса; мы будем требовать, чтобы грамматики не просто перечисляли фразы, но и приписывали каждой фразе одну или несколько структур (последнее — в случае ее синтаксической неоднозначности).

Особый интерес представляет для нас исследование таких алгоритмов, которые для каждой данной фразы решали бы, принадлежит ли она языку, и в случае положительного результата определяли ее синтаксическое дерево (или деревья).

В § 7.2 будут введены строгие определения, формализующие перечисленные интуитивные понятия.

7.1.3. Упражнения

1. Построить грамматику с такими правилами, которые обеспечили бы представление структуры фразы (3) в виде бинарного дерева.
2. Что получится, если ввести правило $Gn \rightarrow Art\ Gn$?

§ 7.2. ОПРЕДЕЛЕНИЯ. РАСПОЗНАВАЕМЫЕ СВОЙСТВА

Прежде всего мы определим некоторый исключительно важный тип грамматик и языков — столь важный, что к нему пришли независимо друг от друга многие исследователи, стоявшие на достаточно различных точках зрения. Однако наиболее четкую и глубокую характеристику этого класса дал Н. Хомский, и потому мы назовем соответствующие языки *языками Хомского*. Сам Хомский назвал их «контекстно-свободными», или «бесконтекстными» (context-free); этот термин объясняется тем, что правила грамматик для контекстно-свободных языков формулируются независимо от какого бы то ни было контекста.

Для обозначения языков Хомского и грамматик Хомского мы будем использовать сокращения «КС-языки» и «КС-грамматики» (КС — от «контекстно-свободный»).

7.2.1. Определения

Грамматика Хомского, или *КС-грамматика*, задается указанием следующих четырех объектов:

. Конечное множество символов V_T — терминальный, или основной, словарь.

.. Конечное множество символов V_A — вспомогательный словарь.

... Аксиома $S \in V_A$.

.... Конечное множество так называемых КС-правил вида $A \rightarrow \varphi$, где

$$A \in V_A, \quad \varphi \in \{V_A \cup V_T\}^*.$$

Таким образом, комбинаторная система, соответствующая грамматике Хомского, является полутуэвской системой, однако специального вида. Эта система имеет «нетерминальные» и, возможно, «терминальные» теоремы[1]). *Языком Хомского* называется множество терминальных теорем полутуэвской системы, иначе говоря, множество фраз над терминальным словарем некоторой грамматики Хомского, порождаемых этой грамматикой.

Применительно к грамматикам Хомского вместо «доказательство» принято говорить «вывод».

Примеры. 1. Пусть имеется грамматика G_m:

$$V_A=\{S\};\quad V_T=\{a, b, c\};$$
$$\left|\begin{array}{l} S\to aSa \\ S\to bSb \\ S\to c \end{array}\right|.$$

Один из возможных выводов в G_m:

$$S\to aSa\to aaSaa\to aabSbaa\to \boldsymbol{aabcbaa}.$$

Мы будем писать

$$S\Rightarrow aabcbaa;$$

запись $\varphi\Rightarrow\psi$ означает, что существует вывод, начинающийся цепочкой φ и заканчивающийся цепочкой ψ.

Ясно, что грамматика G_m порождает в точности те фразы, которые построены по схеме $xm\tilde{x}$, где x — любая цепочка из символов a и b, а $\tilde{x}$ — ее обращение (зеркальный образ). Это можно записать так:

$$L_m=L(G_m)=\{xc\tilde{x}\mid x\in\{a, b\}^*\}.$$

2. Пусть имеется грамматика G_0:

$$V_A=\{S, A\};\quad V_T=\{a, b\};$$
$$\left|\begin{array}{l} S\to aA \\ A\to Ab \end{array}\right|.$$

Очевидно, что выводы в этой грамматике продолжаются бесконечно, не приводя к терминальным цепочкам: G_0 порождает пустой язык.

Классификация правил. Всякая КС-грамматика G порождает некоторое множество нетерминальных цепочек и некоторое (быть может, пустое) множество терминальных цепочек,

[1]) Т. е. теоремы, являющиеся цепочками в $V_A\cup V_T$, но не в V_T (соответственно цепочками в V_T). — *Прим. ред.*

образующее язык $L(G)$. Назовем *предтерминальной* нетерминальную цепочку, из которой может быть получена терминальная применением ровно одного правила; ясно, что предтерминальная цепочка содержит ровно один нетерминальный символ.

Необходимым условием непустоты языка $L(G)$ является наличие в G одного или нескольких *терминальных правил* вида

$$A \to \tau,$$

где $A \in V_A$, $\tau \in V_T^*$.

Все прочие правила грамматики называются *нетерминальными*.

Пример. Грамматика G_m имеет два нетерминальных правила:

$$S \to aSa \quad \text{и} \quad S \to bSb$$

и одно терминальное:

$$S \to c.$$

Грамматика G_0 не имеет ни одного терминального правила.

7.2.2. Соответствие между грамматиками и языками

Таким образом, всякой КС-грамматике соответствует некоторый КС-язык, быть может, пустой. Важно подчеркнуть, что это соответствие не является взаимно однозначным: различные грамматики могут порождать один и тот же язык.

Пример. Пусть имеется грамматика G_n:

$$V_A = \{S, A\}; \quad V_T = \{a, b, c\};$$

$$\left|\begin{array}{ll} S \to aSa & (1) \\ S \to bSb & (2) \\ S \to c & (3) \\ S \to abSba & (4) \\ S \to cA & (5) \\ S \to aA & (6) \end{array}\right| .$$

Грамматика G_n порождает тот же язык L_m, что и G_m, однако отличается от этой последней

. добавлением правила (4), которое обеспечивает «более быстрое» порождение некоторых цепочек языка L_m;

.. добавлением правил (5) и (6), которые вообще не дают никаких терминальных цепочек.

В дальнейшем мы встретимся с содержательно более интересными примерами такого рода.

Итак, пусть дана некоторая грамматика. Мы можем прежде всего, не подвергая ее существенному изменению (т. е. не меняя порождаемого ею языка), произвести над ней следующие преобразования:

1) «очистить» грамматику, выбросив из нее те правила и символы, которые не ведут к терминальным цепочкам;
2) тем или иным образом перестроить ее правила, сохраняя порождаемый ею язык.

Связные грамматики. При рассмотрении грамматики удобно сопоставлять ей граф, вершины которого соответствуют символам нетерминального словаря:

$$S = A_1,\ A_2,\ \ldots,\ A_p,$$

и символам, фигурирующим в терминальных правилах:

$$a_1,\ a_2,\ \ldots,\ a_q.$$

Мы будем проводить дугу из A_i в A_j (соответственно в a_k), если в грамматике имеется правило вида

$$A_i \to \varphi A_j \psi \quad \text{(соответственно } A_i \to a_k\text{)}.$$

Ясно, что если этот граф не связен, то ни одна его компонента связности, не содержащая аксиому, не окажет никакого влияния на порождаемый грамматикой язык, а потому соответствующие правила и символы можно из грамматики удалить.

Таким образом, мы всегда можем предполагать, что рассматриваемая грамматика является связной, т. е. такой, что ее граф связен.

О пустой цепочке. Никакие принципиальные соображения не мешают нам иметь терминальные правила вида

$$A \to E,$$

где E — пустая цепочка.

Однако в техническом отношении оказывается более удобным устранить такие правила. Можно доказать (см. посвященные этому упражнения), что любой КС-грамматике, которая порождает язык L, содержащий пустую цепочку, можно поставить в соответствие «очень близкую к ней» КС-грамматику, порождающую язык $L \setminus \{E\}$. В последующем изложении мы будем предполагать, что эта редукция выполнена.

Сокращение выводов. Пусть в некоторой грамматике имеется вывод $A \Rightarrow \varphi$, где A — нетерминальный символ, а φ — произвольная цепочка. Очевидно, мы не изменим язык, порождаемый этой грамматикой, если добавим к ней правило $A \to \varphi$. Будем говорить, что это правило получено *сокращением вывода*.

7.2.3. *Непустые, конечные и бесконечные языки*

Непустые языки. Необходимым и достаточным условием непустоты языка, порождаемого КС-грамматикой G, является существование вывода $S \Rightarrow \tau$, где τ — терминальная цепочка.

Проблема существования такого вывода разрешима (см. посвященное этому упражнение).

Бесконечные языки. Поскольку терминальный словарь конечен, необходимым условием бесконечности языка $L(G)$ является неограниченность длин выводимых терминальных цепочек. Отсюда следует, что длины выводимых предтерминальных цепочек также не должны быть ограничены; стало быть, не должны быть ограничены и длины выводимых нетерминальных цепочек.

Это последнее условие явно не является достаточным, поскольку возможны грамматики, в которых длины нетерминальных цепочек не ограничены, но которые тем не менее порождают пустой язык[1]).

Тем самым мы можем расчленить проблему бесконечности языка на две, а именно:

а) Найти необходимое и достаточное условие бесконечности множества порождаемых грамматикой нетерминальных цепочек.

б) Найти дополнительное условие бесконечности множества терминальных цепочек.

Чтобы ответить на вопрос а), рассмотрим граф, определенный на стр. 119; при этом будем считать грамматику связной.

Этот граф может содержать цикл или петлю, начинающуюся, скажем, в A и возвращающуюся в A. В этом случае существует вывод вида

$$A \Rightarrow \varphi A \psi.$$

Тривиальный случай $A \Rightarrow A$ соответствует простому повторению A. Исключим из рассмотрения этот случай, положив

$$|\varphi| + |\psi| \neq 0.$$

Поскольку рассматриваемая грамматика является по предположению связной, в ней существует вывод вида

$$S \Rightarrow \alpha A \beta.$$

Следовательно, множество порождаемых грамматикой нетерминальных цепочек («нетерминальный язык») содержит бесконечное множество цепочек вида $\alpha\varphi^n A\psi^n\beta$ и тем самым является бесконечным.

Предположим теперь, что наш граф не содержит ни петель, ни циклов. Тогда правила вида $A_1 \to \varphi_1$ не могут содержать в правых частях A_1. Если правая часть такого правила содержит A_2, то правила вида $A_2 \to \varphi_2$ не могут содержать в правых частях ни A_1, ни A_2 и т. д. Таким образом, нетерминальные символы постепенно исчезают, и после конечного числа шагов их не будет вовсе, так

[1]) Такова, в частности, грамматика G_0 на стр. 117. — *Прим. ред.*

что порождаемый грамматикой «нетерминальный язык» не может быть бесконечным.

Итак, *для того чтобы порождаемый грамматикой нетерминальный язык был бесконечным, необходимо и достаточно, чтобы для некоторого нетерминального символа A существовал вывод*

$$A \Rightarrow \varphi A \psi, \quad |\varphi| + |\psi| \neq 0.$$

Нетрудно доказать, что проблема существования такого нетерминального символа алгоритмически разрешима.

Если $|\varphi| \cdot |\psi| \neq 0$, говорят, что символ A *самовставляющийся*.

Что касается вопроса б), то для него также можно указать разрешимый критерий (см. упр. 7 на стр. 123). Таким образом, проблема распознавания конечности языка является разрешимой. Разрешима и проблема распознавания пустоты (см. упр. 6 на стр. 123). Итак, имеет место

Теорема. *Свойства КС-грамматики порождать пустой язык и порождать конечный язык (а следовательно, и свойства порождать непустой или бесконечный язык) алгоритмически распознаваемы.*

7.2.4. Рекурсивность

Поскольку КС-грамматики имеют довольно специальную форму, можно ожидать, что и порождаемые ими языки являются языками какого-то специального вида.

Так как мы исключили правила вида $A \to E$, на каждом шаге вывода в КС-грамматике длина цепочки может только увеличиваться или в крайнем случае оставаться неизменной. Чтобы ответить на вопрос, принадлежит ли данная цепочка (не обязательно терминальная) языку, порождаемому данной КС-грамматикой, достаточно построить все возможные в этой грамматике выводы, начинающиеся начальным символом и оканчивающиеся цепочками, длины которых меньше или равны длине рассматриваемой цепочки. Указанная процедура применима для любой цепочки и всегда дает ответ «да» или «нет». Короче говоря,

всякий КС-язык является рекурсивным множеством.

7.2.5. Пример языка, не являющегося КС-языком

В дальнейшем нам понадобится тот факт, что язык $L = \{a^n b^n c^n\}$ не является КС-языком. Докажем это утверждение.

Допустим, что L порождается некоторой КС-грамматикой. Поскольку L бесконечен, в этой грамматике существует вывод

$$A \Rightarrow \varphi A \psi, \quad |\varphi| + |\psi| \neq 0.$$

Поэтому в ней выводимы нетерминальные цепочки вида

$$\alpha \varphi^n A \psi^n \beta$$

для любого n. В конце концов φ, A и ψ должны дать терминальные цепочки. Сокращая соответствующие выводы, мы можем прибавить к нашей грамматике правила вида $A \to \tau$.

Легко показать, что, какие бы правила такого вида мы ни выбирали, мы будем получать, помимо цепочек $a^n b^n c^n$, также и другие цепочки, в которых показатели степени при a, b, c не будут равны.

Замечание. Ясно, что язык $L = \{a^n b^n c^n\}$ рекурсивен. Следовательно,

класс КС-языков является собственным подклассом класса рекурсивных языков.

7.2.6. Упражнения

1. Построить КС-грамматики, порождающие языки

$$L_1 = \{a^n b^n c^p\} \quad \text{и} \quad L_2 = \{a^m b^q c^q\}.$$

2. Построить КС-грамматику, порождающую язык $\{x\tilde{x}\}$ над алфавитом $\{a, b\}$ (без метки в середине).

3. Доказать, что язык $\{a^p b^q c^r \mid p \leqslant q \leqslant r\}$ не является КС-языком.

4. Рассмотрим словарь $V = V_A \cup V_T$ и правила вида $A \to \varphi$, $A \in V_A$. Будем брать в качестве аксиом один за другим все элементы словаря V_A. Могут ли при этом порождаться разные языки? Построить простые примеры. Вот один из них:

$$A \to AC$$
$$B \to BC$$
$$A \to Aa$$
$$A \to a$$
$$B \to Bb$$
$$B \to b$$
$$C \to AA$$

5. *Элиминация пустой цепочки.* Пусть G есть КС-грамматика, и пусть V_0 — множество символов $A \in V_A$, таких, что для них имеются правила вида $A \to E$. Образуем множество V_1, добавив к V_0 все $A \in V_A$, такие, что имеются правила вида $A \to \varphi$, где $\varphi \in V_0^*$; аналогично определим V_2 и т. д.

Рассмотреть полученную последовательность множеств. Сформулировать условия, необходимые и достаточные для того, чтобы $E \in L(G)$. Исходя из G, построить КС-грамматику G', такую, что

$$L(G') = L(G) \setminus \{E\}.$$

Заметим, что для этого недостаточно просто устранить все правила вида $A \to E$. В самом деле, рассмотрим грамматику

$$G: \left| \begin{array}{l} S \to aAa \\ A \to E \\ A \to bA \\ A \to c \end{array} \right| .$$

Если выкинуть из G правило $A \to E$, то G не будет порождать ни цепочку aa, ни цепочку $ab^n a$. Ясно, однако, что свойство порождать эти цепочки должно быть сохранено.

6. *Элиминация непродуктивных правил. Пустой язык.* Пусть G есть КС-грамматика. Будем считать, что в G нет правил вида $A \to E$.

Обозначим через W_0 объединение терминального словаря V_T и множества символов A, таких, что $A \in V_A$ и имеются правила вида $A \to \tau$, где τ — терминальная цепочка. Образуем множество W_1, добавив к W_0 все $A \in V_A$, такие, что имеются правила вида

$$A \to \varphi, \quad \text{где} \quad \varphi \in W_0^* \setminus \{E\}.$$

Поступая аналогично, образуем последовательность множеств W_i.

Используя множества W_i, сформулировать условие, необходимое и достаточное для того, чтобы язык $L(G)$ был пуст.

Указать процедуру, позволяющую устранить из грамматики все «непродуктивные» правила.

Заметим, что для грамматики

$$G: \left| \begin{array}{l} S \to BaB \\ B \to Bb \\ B \to b \\ A \to ASA \\ A \to a \end{array} \right|$$

множества W_0 и W_1 суть соответственно $\{A, B\}$ и $\{A, B, S\}$; однако если S — аксиома, то символ A «непродуктивен», поскольку его нельзя получить из S. Дело будет обстоять иначе, если добавить к этой грамматике правило вида $S \to \varphi A \psi$.

7. Используя предыдущее упражнение, указать процедуру, позволяющую распознавать, является ли множество порождаемых грамматикой терминальных цепочек бесконечным.

§ 7.3. СВОЙСТВА ЗАМКНУТОСТИ

Говорят, что множество замкнуто относительно некоторой операции, если результат применения этой операции к любому элементу множества (если операция унарна) или к любой паре

элементов (если операция бинарна) и т. п. содержится в этом множестве. Рассмотрим с этой точки зрения некоторые простые операции над КС-языками.

7.3.1. Объединение КС-языков

Пусть L_1 и L_2 суть КС-языки, заданные соответственно грамматиками G_1 и G_2 с аксиомами S_1 и S_2. Изменив, если нужно, обозначения некоторых или всех вспомогательных символов, можно добиться, чтобы

$$V_{A_1} \cap V_{A_2} = \emptyset.$$

Рассмотрим грамматику с аксиомой

$$S \notin V_{A_1} \cup V_{A_2},$$

правилами которой будут все правила грамматик G_1 и G_2 и сверх того правила $S \to S_1$ и $S \to S_2$.

Ясно, что это КС-грамматика, и она порождает язык $L_1 \cup L_2$ Итак, *объединение двух КС-языков является КС-языком.*

7.3.2. Произведение КС-языков

Сохраняя те же обозначения, определим произведение L_1L_2. Для этого, как и выше, сделаем так, чтобы пересечение вспомогательных словарей грамматик G_1 и G_2 было пусто. Затем возьмем в качестве аксиомы новый символ S, объединим множества правил обеих грамматик и добавим к этому объединению правило $S \to S_1S_2$. Ясно, что таким образом мы построили КС-грамматику, порождающую L_1L_2. Итак, *произведение двух КС-языков является КС-языком.*

7.3.3. Итерация

Только что полученный результат позволяет по КС-грамматике, порождающей язык L, строить КС-грамматики, порождающие языки L^2, L^3 и т. д. Рассмотрим теперь язык

$$L^* \setminus \{E\} = L \cup L^2 \cup L^3 \cup \ldots \cup L^n \cup \ldots.$$

Пусть S — аксиома КС-грамматики G, порождающей L. Добавим к G новый символ T и правила $T \to ST$, $T \to S$; кроме того, объявим T аксиомой вместо S. Новые правила позволяют, очевидно, порождать цепочки $S, S^2, \ldots, S^n, \ldots$, из которых по правилам грамматики G получаются языки $L, L^2, \ldots, L^n, \ldots$. Итак, *итерация КС-языка является КС-языком* (с точностью до пустой цепочки).

7.3.4. Пересечение КС-языков

1. Рассмотрим КС-языки

$$L_0 = \{a^n c a^n\} \quad \text{и} \quad L_m = \{xc\tilde{x} \mid x \in \{a, b\}^*\}$$

(см. п. 7.2.1). Ясно, что $L_0 \cap L_m = L_0$.

2. Рассмотрим КС-языки

$$L_1 = \{a^n b^n c^p\} \quad \text{и} \quad L_2 = \{a^m b^q c^q\}.$$

Имеем $L_1 \cap L_2 = \{a^n b^n c^n\}$; мы видели, что этот язык не является КС-языком. Из приведенных примеров следует, что *пересечение двух КС-языков может быть, но не обязательно является КС-языком.*

7.3.5. Дополнение

Язык V_T^* является КС-языком, поскольку он может быть порожден с помощью грамматики, содержащей всевозможные правила вида $S \to aS$, $S \to Sa$ и $S \to a$.

Под дополнением $\bar{L}$ языка L мы понимаем дополнение относительно V_T^*. Известно, что

$$\overline{L_1 \cap L_2} = \bar{L}_1 \cup \bar{L}_2.$$

Поэтому, если бы операция взятия дополнения сохраняла свойство быть КС-языком, то и пересечение сохраняло бы его; следовательно, *дополнение КС-языка не обязательно является КС-языком.*

7.3.6. Обращение

Операция обращения, примененная к КС-языку L с грамматикой $G: \{A \to \varphi\}$, ставит ему в соответствие язык $\tilde{L}$ с грамматикой $\tilde{G}: \{A \to \tilde{\varphi}\}$. Ясно, что *результат обращения КС-языка является КС-языком.*

7.3.7. Подстановка

Пусть имеется КС-язык L, порождаемый грамматикой G со словарем $V_A \cup V_T$, где $V_T = \{a_i \mid 1 \leqslant i \leqslant n\}$. Пусть, далее, имеются n КС-языков $L_1, \ldots, L_i, \ldots, L_n$ (этих языков столько, сколько символов в V_T), и пусть $G_1, \ldots, G_i, \ldots, G_n$ суть КС-грамматики, порождающие эти языки. Сделаем так, чтобы все попарные пересечения вспомогательных словарей грамматик $G, G_1, \ldots, G_n$ были пусты, и отождествим a_i с аксиомой грамматики G_i, задающей L_i.

Объединение грамматик $G, G_1, \ldots, G_n$ является КС-грамматикой, порождающей язык, получаемый из $L, L_1, \ldots, L_n$ посредством операции, называемой *подстановкой* $L_1, \ldots, L_n$ в L. Итак, *результат подстановки КС-языков в КС-язык является КС-языком.*

Резюмируем: множество КС-языков замкнуто относительно операций объединения, умножения, итерации, обращения и подстановки.

7.3.8. Упражнение

Пусть имеется КС-язык, заданный грамматикой G. Рассмотрим все языки, которые можно получить, поочередно выбирая в качестве аксиомы каждый из нетерминальных символов грамматики G.

Образуем объединение этих языков. Что получится? Попытайтесь сформулировать выводы относительно роли аксиомы в грамматике.

§ 7.4. СПЕЦИАЛЬНЫЕ КЛАССЫ КС-ЯЗЫКОВ

Вводя дополнительные ограничения на вид КС-правил, можно определить различные классы КС-языков. Эти классы вводятся в силу самых разных соображений и не обязательно обладают свойствами замкнутости, присущими классу всех КС-языков. Некоторые свойства таких специальных классов можно вывести непосредственно из алгебраической характеристики КС-языков, которая будет получена ниже.

Поскольку разные КС-грамматики могут порождать один и тот же язык, при классификации языков необходимо соблюдать некоторые предосторожности.

Мы будем определять грамматику того или иного типа (α), формулируя свойства, которыми должны обладать ее правила.

Будем говорить, что язык L относится к типу (α), если существует хотя бы одна грамматика этого типа, порождающая данный язык.

Таким образом, чтобы доказать, что язык *не* относится к типу (α), необходимо показать, что он не может быть порожден **ни** одной грамматикой типа (α).

7.4.1. Автоматные языки

В одной из следующих глав мы подробно будем изучать грамматики, все нетерминальные правила которых имеют вид

$$A \to aB; \quad A,\ B \in V_A,\quad a \in V_T.$$

В литературе на английском языке такие грамматики принято называть finite-state grammars (по-русски — *грамматики с конечным числом состояний*); порождаемые ими языки называют также *регулярными*. Изучение этих языков было начато в известной работе С. К. Клини [1956], в которой была доказана основная теорема о них (см. далее п. 10.4.4).

Мы будем называть грамматики описанного типа *автоматными*, сокращенно *А-грамматиками*[1]) (по причинам, которые выяснятся позднее, в гл. X).

Языки, порождаемые А-грамматиками, мы будем называть *А-языками, или автоматными языками.*

---

[1]) В оригинале «К-грамматики» — от фамилии Kleene. Мы, однако, не сочли возможным отказаться от принятого в русской литературе термина «А-грамматики». — *Прим. ред.*

7.4.2. Линейные языки

Грамматика, все нетерминальные правила которой имеют вид

$$A \to xBy; \quad A, B \in V_A; \quad x, y \in V_T^*,$$

называется *линейной*.

Язык называется *линейным*, если он может быть порожден линейной грамматикой.

А-языки образуют подкласс класса линейных языков. Другой интересный подкласс составляют *минимальные линейные языки*, для которых V_A состоит из одного символа S, а множество терминальных правил — из одного правила $S \to c$, причем символ c не встречается ни в каком другом правиле. Язык L_m из примера на стр. 117 является минимальным линейным языком.

7.4.3. Металинейные языки

КС-грамматика называется *металинейной*, если правые части ее правил не содержат аксиомы и все правила, левые части которых отличны от аксиомы, имеют такой же вид, как правила линейной грамматики.

Язык называется *металинейным*, если существует порождающая его металинейная грамматика.

Пример. Рассмотрим язык, заданный словарями $V_A = \{S, T\}$, $V_T = \{a, b, c\}$ и грамматикой

$$G_{2m} : \left| \begin{array}{l} S \to TT \\ T \to aTa \\ T \to bTb \\ T \to c \end{array} \right|;$$

G_{2m} — металинейная грамматика. Она порождает «двойной зеркальный язык» $L_{2m} = \{xc\tilde{x}yc\tilde{y}\}$.

Контрпример. Грамматика

$$\left| \begin{array}{l} S \to aSS \\ S \to b \end{array} \right|$$

не является металинейной. Порождаемый ею язык состоит из «скелетов» всевозможных выражений, образованных из предметных символов и символов бинарных операций в так называемой бесскобочной записи (записи Лукасевича). Можно доказать, что этот язык не порождается никакой металинейной грамматикой, т. е. не является металинейным.

7.4.4. Последовательностные грамматики

Грамматика называется *последовательностной*, если ее нетерминальные символы можно расположить в последовательность $A_1, A_2, \ldots, A_k$ таким образом, что правила с левой частью A_1 не содержат в правых частях нетерминальных символов, отличных от A_1, правила с левой частью A_2 не содержат нетерминальных символов, отличных от A_1 и A_2, и т. д.

Язык называется *последовательностным*, если он может быть порожден последовательностной грамматикой.

Примеры. Язык $L_m = \{xc\tilde{x}\}$ тривиальным образом является последовательностным, как и всякий язык, порождаемый КС-грамматикой с одним нетерминальным символом.

Язык L_{2m} также является последовательностным: в грамматике G_{2m} можно положить $A_1 = T$, $A_2 = S$.

Напротив, грамматика

$$\left|\begin{array}{l} S \to adTda \\ S \to aSa \\ S \to aca \\ T \to bdSdb \\ T \to bTb \end{array}\right|$$

не является последовательностной; можно показать, что и порождаемый ею язык — не последовательностный.

7.4.5. Другие классы КС-грамматик и КС-языков

Исходя из потребностей описания синтаксической структуры словосочетаний и фраз естественных языков, многие авторы вводили различные специальные классы КС-грамматик.

. В нормальных грамматиках все нетерминальные правила имеют вид

$$A \to BC,$$

где $A \in V_A$, B, $C \in V_A \cup V_T$. Вывод в таких грамматиках задает бинарное синтаксическое дерево фразы (точнее, бинарное дерево составляющих). Такие грамматики соответствуют стремлению многих сторонников теории непосредственно составляющих к дихотомическому анализу фразы. Терминальные правила нормальных грамматик имеют вид $A \to a$, $a \in V_T$; эти правила соответствуют словарю языка.

.. В *предсказуемостных грамматиках* все правила имеют вид

$$A \to a\psi, \quad \text{где} \quad a \in V_T, \quad \psi \in V_A^*.$$

... В *грамматиках зависимостей*[1]) все правила имеют вид

$$A \to \varphi a \psi, \quad \text{где} \quad a \in V_T, \quad \varphi, \psi \in V_A^*.$$

7.4.6. Упражнения

1. Доказать, что язык $\{a^m b^m a^n b^n\}$ не является линейным.
2. Какие свойства замкнутости имеют место для линейных языков?
3. Что можно сказать о произведении линейного языка и А-языка?
4. Какие свойства замкнутости имеют место для металинейных языков?

§ 7.5. УПРАЖНЕНИЯ

1. Доказать, что язык, порождаемый грамматикой, приведенной в п. 7.4.4, — не последовательностный.
2. Доказать, что язык $\{xc\tilde{x}cyc\tilde{y}\}$ не является линейным.
Доказать, далее, что язык $\{\tilde{x}xy\tilde{y}\}$ — также не линейный.
Сравнить методы доказательства и объяснить, почему во втором случае не удается применить сравнительно простой метод, приводящий к цели в первом случае.
3. Построить пример КС-языка, не являющегося А-языком и порождаемого КС-грамматикой, каждое правило которой является либо леволинейным, либо праволинейным. (Правило называется *леволинейным*, соответственно *праволинейным*, если оно имеет вид $A \to xB$, соответственно $A \to Bx$, где $B \in V_A$, $x \in V_T^*$.) Показать, что всякий линейный язык может быть порожден такой грамматикой.

---

[1]) Грамматика зависимостей позволяет сопоставить каждой цепочке порождаемого ею языка дерево зависимостей (дерево синтаксического подчинения) следующим образом. Пусть на некотором шаге вывода терминальной цепочки применяется правило $A \to B_1 \ldots B_k a B_{k+1} \ldots B_s$, где $a \in V_T$, $B_1, \ldots, B_s \in V_A$; тогда на каких-то следующих шагах должны применяться правила вида $B_1 \to \varphi_1 b_1 \psi_1$, ..., $B_s \to \varphi_s b_s \psi_s$, где $b_1, \ldots, b_s \in V_T$, $\varphi_1, \ldots, \varphi_s$, $\psi_1, \ldots, \psi_s \in V_A^*$. В этом случае в соответствующей терминальной цепочке из (данного вхождения) a проводятся стрелки в (данные вхождения) $b_1, \ldots, b_s$. Подробнее о грамматиках зависимостей см. Фитиалов [1968]. — *Прим. ред.*

Глава VIII

НЕРАСПОЗНАВАЕМЫЕ СВОЙСТВА КС-ГРАММАТИК

Напомним, что свойство объектов определенного класса называется (алгоритмически) распознаваемым, если существует общий алгоритм, позволяющий для любого конкретного объекта этого класса ответить на вопрос, обладает ли он данным свойством; свойство называется (алгоритмически) нераспознаваемым, если такого алгоритма не существует. Чтобы установить нераспознаваемость некоторого свойства для данного класса C, достаточно установить его нераспознаваемость для какого-либо подкласса класса C.

В предыдущей главе были рассмотрены некоторые распознаваемые свойства КС-грамматик, а именно свойство порождать пустой, конечный и бесконечный язык, а также свойство содержать данную цепочку. Теперь мы рассмотрим некоторые нераспознаваемые свойства КС-грамматик.

§ 8.1. ПРОБЛЕМЫ, СВЯЗАННЫЕ С ПЕРЕСЕЧЕНИЕМ

8.1.1. Проблема непустоты пересечения

Рассмотрим прежде всего следующую проблему:

(1) Построить алгоритм, который для произвольной пары КС-грамматик позволял бы установить, является ли пересечение порождаемых ими языков пустым.

Докажем, что эта проблема неразрешима. Для этого достаточно показать, что она неразрешима для какого-то частного случая.

Рассмотрим два специальных КС-языка в алфавите $\{a, b, c\}$:

1) Язык L'_m, отличающийся от языка L_m из п. 7.2.1 только тем, что он не содержит цепочки 'c'. Этот язык порождается грамматикой

$$G'_m : \left| \begin{array}{l} S \to aSa \\ S \to bSb \\ S \to aca \\ S \to bcb \end{array} \right|.$$

2) Язык L_n, порождаемый грамматикой

$$G_n : \left| \begin{array}{ll} S \to g_i S h_i, & 1 \leqslant i \leqslant n \\ S \to c & \end{array} \right|,$$

где g_i и h_i — непустые цепочки в алфавите $\{a, b\}$.

Язык L_n содержит цепочку 'c'; именно поэтому мы изменили язык L_m так, чтобы он не содержал этой цепочки — иначе пересечение этих двух языков всегда было бы непусто.

Вопрос о том, пусто ли пересечение языков L_m и L_n, равносилен вопросу, содержит ли L_n симметричную цепочку $Mc\tilde{M}$. Иначе говоря, он равносилен вопросу о существовании последовательности индексов

$$i_1, i_2, \ldots, i_k,$$

такой, что цепочка

$$g_{i_1} g_{i_2} \cdots g_{i_k}$$

есть обращение цепочки

$$h_{i_k} \ldots h_{i_2} h_{i_1}.$$

В конечном счете нужно выяснить, существует ли такая последовательность индексов $i_1, i_2, \ldots, i_k$, что $g_{i_1} g_{i_2} \ldots g_{i_k}$ совпадает с $\tilde{h}_{i_1} \tilde{h}_{i_2} \ldots \tilde{h}_{i_k}$.

Нетрудно узнать в этой задаче проблему Поста для пар

$$(g_1, \tilde{h}_1), \ldots, (g_n, \tilde{h}_n);$$

следовательно, поставленная нами проблема неразрешима.

Поскольку проблема (1) неразрешима для данного частного случая (а именно, для пар с фиксированным первым элементом G'_m и вторым элементом вида G_n, зависящим от выбора цепочек $g_1, \ldots, g_k, h_1, \ldots, h_k$), она неразрешима и в общем случае.

8.1.2. Проблема автоматности пересечения

Если пересечение определенных выше языков L'_m и L_n содержит цепочку

$$g_{i_1} g_{i_2} \cdots g_{i_k} c h_{i_k} \cdots h_{i_2} h_{i_1} = xc\tilde{x}$$

(соответствующую решению проблемы Поста), то оно содержит также для любого n цепочку

$$(g_{i_1} g_{i_2} \cdots g_{i_k})^n c\, (h_{i_k} \cdots h_{i_2} h_{i_1})^n.$$

Таким образом, пересечение двух КС-языков оказывается *бесконечным* подмножеством языка L'_m. Однако ни одно бесконечное подмножество этого языка не может быть А-языком (это легко показать, используя один из основных фактов теории А-языков, который будет установлен ниже, а именно следствие из теоремы п. 14.3.5). В то же время пустой язык, очевидно, является автоматным. Итак, пересечение $L'_m \cap L_n$ является А-языком тогда и только тогда, когда оно пусто. Но проблема пустоты пересечения $L'_m \cap L_n$ неразрешима; следовательно, неразрешима и проблема его

автоматности. Тем более неразрешима проблема автоматности пересечения двух КС-языков в общем случае. Таким образом, мы можем сформулировать следующую теорему.

Теорема. *Не существует алгоритмов, позволяющих установить*: а) *является ли пересечение двух КС-языков непустым*; б) *является ли пересечение двух КС-языков А-языком.*

Эта теорема остается справедливой даже для такого специального случая, как пересечение двух минимальных линейных языков, один из которых фиксирован.

8.1.3. Проблема «контекстной свободности» пересечения

С помощью аналогичной техники, используя вместо L'_m язык $L_{2m} = \{xc\tilde{x}cyc\tilde{y}\}$, а вместо L_n — язык, порождаемый грамматикой

$$\left|\begin{array}{ll} S \to aSa & \\ S \to bSb & \\ S \to \acute{a}cTca & \\ S \to bcTcb & \\ T \to g_i T h_i, & 1 \leqslant i \leqslant n \\ T \to c & \end{array}\right|,$$

и замечая, что никакой бесконечный подъязык языка $\{xc\tilde{x}cxc\tilde{x}\}$ не является КС-языком (это устанавливается рассуждениями, сходными с теми, которые были применены в п. 7.2.5), можно доказать следующую теорему:

Теорема. *Не существует алгоритма, позволяющего установить, является ли пересечение двух КС-языков КС-языком.*

8.1.4. Дополнение

Нетрудно было бы показать (непосредственным построением соответствующих грамматик), что дополнения $\bar{L}_m$ и $\bar{L}_n$ языков L_m и L_n являются КС-языками. Поэтому $\bar{L}_m \cup \bar{L}_n$ — также КС-язык. Но $\overline{\bar{L}_m \cup \bar{L}_n} = L_m \cap L_n$, а мы доказали, что проблемы распознавания пустоты, конечности и автоматности пересечения $L_m \cap L_n$ неразрешимы; таким образом, не существует алгоритмов, распознающих пустоту, конечность или автоматность дополнения к $\bar{L}_m \cup \bar{L}_n$. То же рассуждение можно применить к языкам, использованным в п. 8.1.3. Все это позволяет сформулировать следующую теорему:

Теорема. *Свойства дополнения КС-языка быть пустым, конечным, А-языком и КС-языком нераспознаваемы.*

Эквивалентность. Из только что сформулированной теоремы следует, что не существует алгоритма, позволяющего устанавливать эквивалентность двух произвольных КС-грамматик G_1

и G_2. В самом деле, если бы такой алгоритм существовал, он позволял бы для произвольной КС-грамматики G отвечать на вопрос, верно ли, что $L(G)=V_T^*$, а это равносильно ответу на вопрос, является ли $\overline{L(G)}$ пустым.

Точно так же неразрешима проблема включения, т. е. не существует алгоритма, который для двух КС-языков L_1 и L_2 отвечал бы «да» или «нет» на вопрос: верно ли, что $L_1 \subset L_2$ (действительно, такой алгоритм отвечал бы и на вопрос: верно ли, что $V_T^* \subset L(G)$, т. е. пусто ли $\overline{L(G)}$)?

§ 8.2. ПРОБЛЕМЫ, СВЯЗАННЫЕ С НЕОДНОЗНАЧНОСТЬЮ

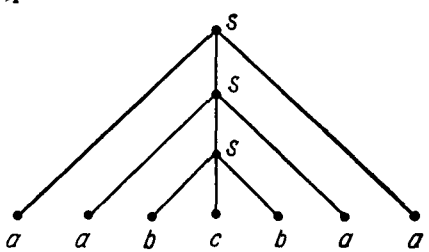
В первом параграфе предыдущей главы мы показали, как можно описать структуру фразы с помощью дерева. КС-грамматики предназначаются для формализации именно этого способа представления синтаксической структуры. Правила вида $A \to \varphi$, где в левой части заменяется *ровно один* символ, позволяют представить любой вывод либо в виде дерева с помеченными узлами, либо с помощью помеченных скобок. Правда, в таком представлении не отражается порядок выполнения подстановок, однако он не имеет существенного значения.

8.2.1. Деревья вывода

Вывод цепочки '*aabcbaa*' в грамматике

$$G'_m : \left| \begin{array}{l} S \to aSa \\ S \to bSb \\ S \to c \end{array} \right|$$

представляется деревом



или эквивалентной скобочной записью

$$\underset{S}{[}a\,\underset{S}{[}a\,\underset{S}{[}b\,\underset{S}{[}c\,\underset{S}{]}\,b\,\underset{S}{]}\,a\,\underset{S}{]}\,a\,\underset{S}{]},$$

которая могла бы быть порождена непосредственно с помощью слегка видоизмененной грамматики G'_m, содержащей два новых терминальных символа '$\underset{S}{[}$' и '$\underset{S}{]}$', а именно с помощью грамматики

$$G''_m : \left| \begin{array}{l} S \underset{S}{\rightarrow} [aSa] \\ S \underset{S}{\rightarrow} [bSb] \\ S \underset{S}{\rightarrow} [c] \end{array} \right|.$$

На этом примере еще нельзя увидеть, какие преимущества дает использование пометок при скобках, поскольку в G''_m имеется только одна пометка. По существу мы привели этот пример только для того, чтобы иметь возможность сопоставить его со следующим.

8.2.2. Нормальная грамматика, порождающая L_m

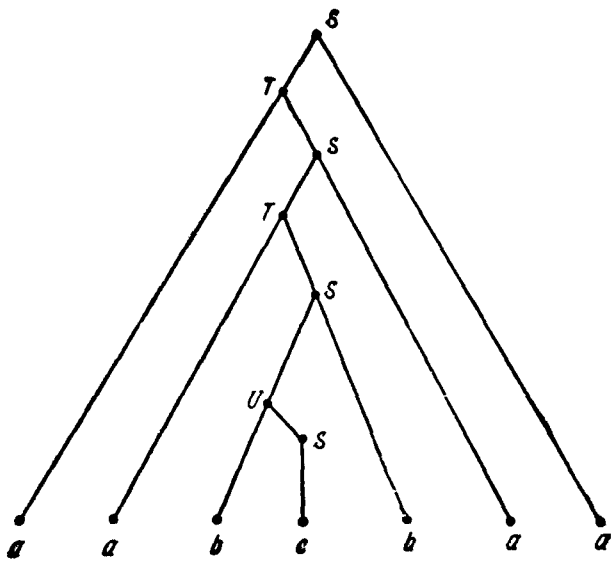
Рассмотрим вывод той же самой цепочки в некоторой нормальной грамматике, также порождающей язык L_m.

Пусть дана нормальная грамматика

$$V_A = \{S, T, U\};\ V_T = \{a, b, c\};\ \text{аксиома } S;$$

$$G^+_m : \left| \begin{array}{l} S \rightarrow Ta \\ T \rightarrow aS \\ S \rightarrow Ub \\ U \rightarrow bS \\ S \rightarrow c \end{array} \right|.$$

Вывод цепочки '*aabcbaa*' в этой грамматике дает следующее дерево:



Скобочная запись, соответствующая этому дереву, такова:

$$\underset{ST}{[}\,\underset{ST}{[a}\,\underset{ST}{[}\,\underset{SU}{[a}\,\underset{SSU}{[}\,\underset{ST}{[b}\,\underset{STS}{[c]}\,\underset{ST}{]b]}\,\underset{STS}{]a]}\,\underset{STS}{]a]}.$$

Пометки при правых скобках можно и не писать, поскольку они однозначно определяются по соответствующим левым пометкам. При таком упрощении приведенное выражение примет вид

$$\underset{ST}{[}\,\underset{ST}{[a}\,\underset{ST}{[}\,\underset{SU}{[a}\,\underset{S}{[}\,\underset{ST}{[b}\,\underset{S}{[c]}\,\underset{ST}{]b]}\,\underset{S}{]a]}\,\underset{S}{]a]}.$$

В некоторых случаях можно даже вводить соглашения, позволяющие вообще опускать правые скобки.

8.2.3. Неоднозначность[1])

Будем говорить, что цепочка *неоднозначна относительно данной грамматики*, если она может быть получена в этой грамматике с помощью по крайней мере двух выводов, изображаемых разными деревьями. Количество разных выводов (точнее, деревьев), возможных в данной грамматике для данной цепочки, характеризует «степень неоднозначности» этой цепочки.

Ясно, что неоднозначность должна определяться относительно *одной* грамматики, поскольку разные, но эквивалентные грамматики (как, например, G_m^+ и G_m'' в предыдущем параграфе), порождая один и тот же язык, могут сопоставлять одинаковым цепочкам разные синтаксические структуры.

Пример. Рассмотрим грамматику

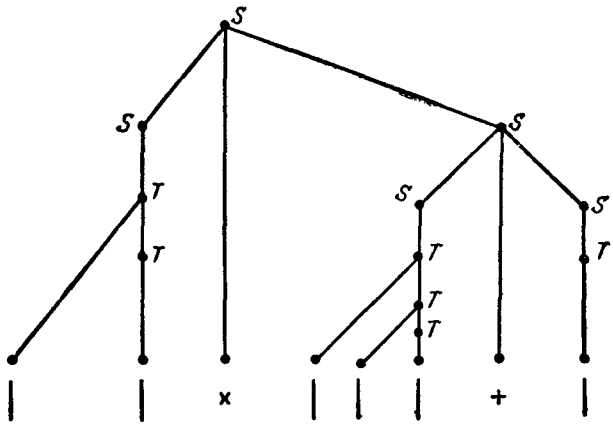
$$(C)\ :\ \left|\begin{array}{l} S \to S \times S \\ S \to S + S \\ S \to T \\ T \to |T \\ T \to | \end{array}\right|.$$

Цепочки, порождаемые этой грамматикой, могут интерпретироваться следующим образом. Палочки (|) служат для записи натуральных чисел: $|=1$, $||=2$, $|||=3$ ...; знаки $\times$ и $+$ имеют обычный арифметический смысл.

---

[1]) В некоторых оригинальных и переводных работах на русском языке (в частности, в переводе книги Гинзбурга [1966]) вместо *неоднозначный, неоднозначность* (фр. *ambigu, ambiguïté*, англ. *ambiguous, ambiguity*) употребляются термины *неопределенный, неопределенность*. Такое словоупотребление следует признать неудачным: оно расходится с лингвистической терминологией (соответствующее явление в естественных языках принято называть синтаксической неоднозначностью) и в чисто математическом контексте также ведет к известным неудобствам. Пишущий эти строки вынужден отметить, что именно он был инициатором введения этих неудачных терминов. — *Прим. ред.*

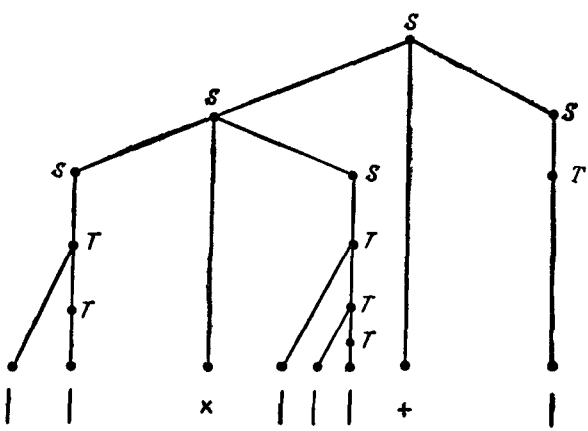
Рассмотрим цепочку $||\times|||+|$. Ее структуру можно описать либо деревом



которое отвечает интрепретации

$||\times(|||+|)$, что дает число 8,

либо деревом



которому соответствует

$(||\times|||)+|$, т. е. 7.

Подобные неоднозначности могут возникать, в частности, в языках программирования; чтобы избежать их, прибегают к следующим методам:

. Упорядочивают символы операций по старшинству; это по существу означает упорядочение правил грамматики и введение порядка в процесс анализа заданной цепочки.

.. Вводят дополнительные терминальные символы (например, скобки в только что разобранном примере).

... Используют бесскобочную запись, например:

$$(C_p) \quad : \quad \left| \begin{array}{l} S \to \times SS \\ S \to + SS \\ S \to T \\ T \to |T \\ T \to |. \end{array} \right| .$$

Точка в правой части последнего правила вводится в качестве маркера, разделяющего записи чисел; без такого маркера конкатенация цепочек, состоящих из палочек, приводила бы к неоднозначности.

В бесскобочной записи разные интерпретации приведенной выше цепочки выглядят по-разному:

$$(1_p) \times ||.+|||.|.$$
$$(2_p) + \times ||.|||.|.$$

Определения. Грамматика, порождающая язык, содержащий неоднозначные относительно нее цепочки, называется *неоднозначной*.

Если некоторая цепочка имеет в данной грамматике p разных (т. е. представляемых разными деревьями) выводов, мы говорим, что *степень ее неоднозначности* равна p.

8.2.4. *Нераспознаваемость неоднозначности*

Возникает следующий вопрос: существует ли алгоритм, который для любой КС-грамматики определял бы, является ли она неоднозначной?

Покажем, что эта проблема неразрешима уже для случая линейных грамматик с тремя вспомогательными символами.

Рассмотрим n пар цепочек в алфавите $\{a, b\}$:

$$(g_1, h_1), \ldots, (g_i, h_i), \ldots, (g_n, h_n).$$

Рассмотрим также n различных маркеров $d_1, d_2, \ldots, d_n$ и две минимальные линейные грамматики с центральным маркером c:

$$G_1 \quad : \quad \left| \begin{array}{l} \{S_1 \to d_i S_1 g_i \mid 1 \leqslant i \leqslant n\} \\ S_1 \to c \end{array} \right| ,$$

$$G_2 \quad : \quad \left| \begin{array}{l} \{S_2 \to d_i S_2 h_i \mid 1 \leqslant i \leqslant n\} \\ S_2 \to c \end{array} \right| .$$

Ни та, ни другая грамматика не являются неоднозначными, поскольку в каждой из них все порождаемые ими цепочки различаются между собой наличием или порядком маркеров.

Проблема пустоты пересечения языков $L(G_1)$ и $L(G_2)$ оказывается по существу проблемой Поста: действительно, она равносильна вопросу о возможности равенства

$$d_{i_1} \ldots d_{i_k} c g_{i_1} \ldots g_{i_k} = d_{i_1} \ldots d_{i_k} c h_{i_1} \ldots h_{i_k}$$

для какой-нибудь последовательности индексов $i_1, \ldots, i_k$.

Рассмотрим теперь объединение языков $L(G_1)$ и $L(G_2)$; оно порождается грамматикой $G_{1,2}$ с аксиомой S:

$$G_{1,2}: \left| \begin{array}{l} S \to S_1 \\ S \to S_2 \\ G_1 \\ G_2 \end{array} \right|.$$

Цепочка f является неоднозначной тогда и только тогда, когда ее можно вывести двумя способами, а именно

$$S \to S_1 \to (\text{через } G_1) \to f$$

и

$$S \to S_2 \to (\text{через } G_2) \to f;$$

узнать, возможны ли для какой-нибудь цепочки оба способа — это опять проблема Поста.

Таким образом, проблема неоднозначности неразрешима уже для линейных грамматик с тремя вспомогательными символами. Отсюда вытекает и ее неразрешимость в общем виде.

Теорема. *Проблема неоднозначности КС-грамматик неразрешима.*

8.2.5. Существенная неоднозначность

Рассмотрим грамматику

$$G_a: \left| \begin{array}{l} S \to aSa \\ S \to bSb \\ S \to aaSaa \\ S \to c \end{array} \right|.$$

Относительно этой грамматики всякая цепочка, содержащая два последовательных вхождения символа a, неоднозначна: такую цепочку можно получить либо двукратным применением первого правила, либо однократным применением третьего.

Напротив, грамматика G'_m из п. 8.2.1 порождает тот же самый язык без неоднозначности. Мы говорим, что язык $L_m [= L(G'_m) =$

$= L(G_a)$] не является *неоднозначным* (или *существенно неоднозначным*) *относительно класса минимальных линейных грамматик.*

Для любого класса КС-грамматик можно поставить вопрос: существует ли такой язык, что всякая порождающая его грамматика, принадлежащая к этому классу, является неоднозначной (т. е. существует ли язык, существенно неоднозначный относительно данного класса)?

Р. Парик [1961] показал, что любая КС-грамматика, порождающая язык

$$L_{la} = \{a^p b^q a^r b^s \mid p = r \text{ или } q = s\} = \{a^n b^q a^n b^s\} \cup \{a^p b^m a^r b^m\},$$

является неоднозначной. (Именно, в любой КС-грамматике, порождающей этот язык, найдутся неоднозначные цепочки вида $a^n b^n a^n b^n$.) [1])

§ 8.3. СУЩЕСТВЕННАЯ НЕОДНОЗНАЧНОСТЬ МИНИМАЛЬНЫХ ЛИНЕЙНЫХ ЯЗЫКОВ

Цель настоящего параграфа — изучить явление существенной неоднозначности применительно к одному простому классу КС-грамматик, а именно к минимальным линейным грамматикам. Для более широких классов КС-грамматик такое изучение было бы значительно сложнее.

8.3.1. Пример существенно неоднозначного минимального линейного языка

Рассмотрим язык

$$L_1 = \{a^m c a^n \mid m \geqslant n \geqslant 0\}.$$

Напомним, что линейная грамматика называется минимальной, если она имеет только один нетерминальный символ — начальный — и только одно терминальное правило вида $S \to c$, причем символ c встречается только в этом правиле.

Докажем, что всякая минимальная линейная грамматика, порождающая язык L_1, является неоднозначной.

Любая минимальная линейная грамматика, порождающая L_1, будет иметь вид

$$\left| \begin{array}{l} \{S \to a^{p_i} S a^{q_i} \mid 1 \leqslant i \leqslant N\} \\ S \to c \end{array} \right|.$$

Ясно, что $p_i \geqslant q_i$: в противном случае грамматика будет порождать некоторые цепочки вида $a^m c a^n$, где $m < n$. Поэтому каждое

[1]) Проблема распознавания существенной неоднозначности КС-языка относительно класса всех КС-грамматик алгоритмически неразрешима (см. Гладкий [1965], Гинзбург [1966, гл. VI], Грейбах [1968]). — *Прим. ред.*

правило грамматики будет принадлежать к одному из трех следующих типов:

$$S \to a^{p_k} S, \quad p_k > 0,$$
$$S \to a^{p_j} S a^{q_j}, \quad p_j \geqslant q_j > 0,$$
$$S \to c.$$

Поскольку эти правила должны давать, в частности, цепочки ac и aca, наша грамматика обязательно содержит правила $S \to aS$, $S \to aSa$.

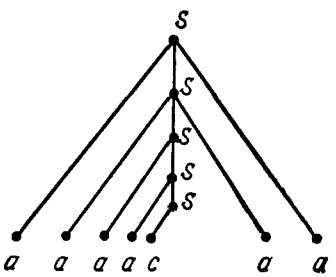
С другой стороны, ясно, что этих правил вместе с правилом $S \to c$ достаточно для порождения L_1. Посмотрим, как ведет себя содержащая их грамматика.

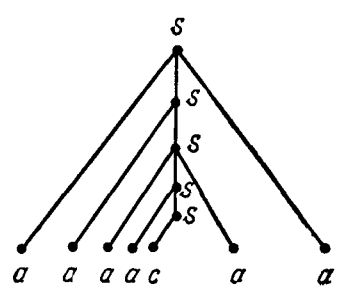
Пусть, например, мы хотим получить цепочку $a^m c a^n$. Применим λ раз правило $S \to aS$ и μ раз правило $S \to aSa$; имеем $\lambda + \mu = m$, $\mu = n$.

Дерево вывода зависит от того, в каком порядке применялись правила, но различных последовательностей применений правил существует столько, сколькими способами можно выбрать n элементов из m элементов, т. е. C_m^n.

Итак, степень неоднозначности цепочки $a^m c a^n$ равна C_m^n.

Пример. Для цепочки $a^4 c a^2$ мы имеем $C_4^2 = 6$ выводов, а именно





и т. д.

Заметим, что L_1 не является существенно неоднозначным относительно класса всех линейных КС-грамматик. Действительно, приводимая ниже грамматика G_L порождает его без неоднозначности:

$$G_L : \left| \begin{array}{l} S \to aSa \\ S \to aT \\ T \to aT \\ T \to c \\ S \to c \end{array} \right| .$$

Заметим, что грамматика G_L — последовательностная.

Итак, *язык может быть существенно неоднозначным относительно некоторого класса грамматик и не быть таковым относительно другого класса.*

8.3.2. Достаточное условие однозначности минимальной линейной грамматики

Пусть $\mathfrak{A}^*$ — свободная полугруппа с базой $\mathfrak{A} = \{x_1, \ldots, x_n\}$ и $\mathfrak{B}^*$ — свободная полугруппа с базой $\mathfrak{B} = \{a, b, \ldots, l\}$. Рассмотрим следующее отображение $\mathfrak{A}^*$ в (или, быть может, на) $\mathfrak{B}^*$: каждое вхождение буквы x_i в цепочках множества $\mathfrak{A}^*$ заменяется на определенную цепочку f_i из $\mathfrak{B}^*$. Такое отображение называется *кодированием* (а образ цепочки при этом отображении — *кодом*), если каждая из получаемых цепочек в алфавите $\mathfrak{B}$ допускает только одну интерпретацию (т. е. только одно разложение на образы элементов $\mathfrak{A}$).

Пример. $\mathfrak{A} = \{1, 2, 0\}$; $\mathfrak{B} = \{a, b\}$;

$$\left.\begin{array}{l} 1 \to ba \\ 2 \to baa \\ 0 \to b \end{array}\right\} \text{— кодирование;}$$

$$\left.\begin{array}{l} 1 \to a \\ 2 \to aa \\ 0 \to b \end{array}\right\} \text{— не кодирование,}$$

поскольку $11 \to aa$ и $2 \to aa$.

Теперь ясно, что минимальная линейная грамматика

$$\left| \begin{array}{l} \{S \to f_i S g_i \mid 1 \leqslant i \leqslant N\} \\ S \to c \end{array} \right|$$

будет однозначной, если хотя бы одно из двух отображений

$$\{i \to f_i;\ i \to g_i \mid 1 \leqslant i \leqslant N\}$$

является кодированием. Из определения видно, что вывод любой цепочки в такой грамматике восстанавливается по ней однозначно.

Заметим, что сформулированное условие не является необходимым. Может оказаться, что ни одно из указанных отображений не есть кодирование и тем не менее всякий вывод определяется однозначно.

Пример. Пусть имеется грамматика

$$\left|\begin{array}{l} S \to aaSa \\ S \to bbSb \\ S \to aSbb \\ S \to bSaa \\ S \to c \end{array}\right|.$$

Множество цепочек f_i, а именно $\{aa, bb, a, b\}$, совпадает с множеством g_i и не дает кодирования. Однако грамматика однозначна: достаточно рассмотреть одновременно первую и последнюю букву произвольной цепочки, чтобы однозначно установить, какое правило было применено первым, вторым и т. д.

Глава IX

АВТОМАТЫ С МАГАЗИННОЙ ПАМЯТЬЮ

В § 7.1 главы VII мы показали, что класс рекурсивно перечислимых языков связан с классом машин Тьюринга; однако для описания естественных языков целесообразно рассмотреть языки более частного вида.

Таким более частным классом языков являются, например, контекстно-свободные языки. Они связываются не с самым общим классом комбинаторных систем (эквивалентным классу произвольных машин Тьюринга), а с некоторым весьма специальным классом полутуэвских систем. Имеется и специальный класс автоматов, соответствующий контекстно-свободным языкам. Это так называемые автоматы с магазинной памятью (англ. push-down store automata).

Мы не сможем теперь ограничиваться рассмотрением лишь детерминированных автоматов, как мы поступали до сих пор; начиная с этой главы, слово «автомат» будет использоваться в качестве сокращения для выражения «недетерминированный (точнее, не обязательно детерминированный) автомат».

Строгое доказательство эквивалентности автоматов с магазинной памятью и КС-грамматик можно найти у Хомского [1963] или Гинзбурга [1966, § 2.5]; ср. также § 15.3.

§ 9.1. АВТОМАТЫ, ДОПУСКАЮЩИЕ ЯЗЫКИ

Излагаемая ниже концепция автомата представляет собой формализацию хорошо известного программистам понятия «магазинная память» (англ. push-down stack/store).

9.1.1. Содержательное описание автоматов с магазинной памятью

Автомат с магазинной памятью (сокращенно: МП-автомат) — это система, состоящая из следующих трех компонентов:

1) *Входная лента* E, разделенная на ячейки и потенциально неограниченная в обе стороны.

2) *Рабочая лента*, или *лента* (*магазинной*) *памяти* M, ограниченная с одной стороны (условимся считать, что слева), но потенциально неограниченная с другой; она также разделена на ячейки.

3) *Управляющее устройство* A, имеющее две головки:

- *входная головка* читает символы на входной ленте E;

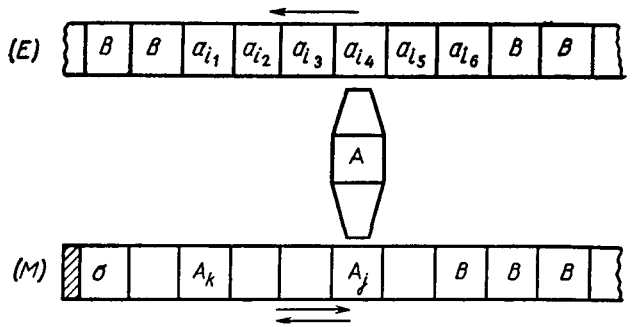
.. *рабочая головка* может читать символы на рабочей ленте M, а также производить на ней стирание и запись символов.

Каждая из головок может передвигать свою ленту определенным образом; каким именно, будет уточнено ниже.

Входная лента. До начала работы МП-автомата на входной ленте может быть записана некоторая информация — по одному символу в каждой ячейке. Для записи этой информации используется *входной словарь* (*входной алфавит*)

$$V_E = \{a_1, a_2, \ldots, a_p\}.$$

Из соображений удобства к этому словарю присоединяется *единичный символ* e, имеющий особый статус (см. ниже). В результате мы получаем:



Единичный символ e не следует смешивать с пробелом (пустым символом), который обозначается через B. Цепочки, составленные из символов входного алфавита V_E, записываются на ленте E так, что слева и справа остаются пробелы. Запись осуществляется слева направо.

Рабочая лента. Запись на рабочей ленте осуществляется самим автоматом с помощью *рабочего словаря* (*рабочего алфавита*)

$$V_M = \{A_1, A_2, \ldots, A_q\}.$$

Этот словарь может включать и V_E. К словарю V_M мы присоединим единичный символ e и еще один выделенный символ σ, который не используется автоматом в ходе его работы. Получаем словарь

$$\bar{V}_M = \{e\} \cup V_M \cup \{\sigma\}.$$

Управляющее устройство. Управляющее устройство способно принимать конечное число *внутренних состояний* $S_0, S_1, \ldots, S_i, \ldots, S_j, \ldots, S_n$.

9.1.2. Функционирование МП-автомата

Для описания функционирования МП-автомата нам понадобятся следующие понятия:

1) Функция $\lambda(\varphi)$, определенная на $\overline{V}_M^*$ следующим образом:

$$\lambda(\sigma) = -1,$$
$$\lambda(e) = 0,$$
$$\lambda(\varphi) = |\varphi|, \quad \text{если} \quad \varphi \in V_M^*.$$

2) *Физическая ситуация* автомата — это тройка

$$\{a_i, S_j, A_k\},$$

где a_i — символ, записанный на ленте E и обозреваемый входной головкой, S_j — состояние автомата и A_k — символ, записанный на ленте M и обозреваемый рабочей головкой.

3) *Ситуация* автомата — это либо физическая ситуация автомата, либо тройка, полученная из некоторой физической ситуации замещением символа a_i или A_k (или обоих) символом e. Именно здесь проявляется недетерминированный характер МП-автомата: в некоторой данной физической ситуации автомат может «закрыть» тот «глаз», которым он обозревал a_i, или тот, которым он обозревал A_k; тогда он «видит» этим «глазом» единичный символ.

Если автомат находится в физической ситуации $\{a_i, S_j, A_k\}$, то он одновременно находится в следующих четырех ситуациях:

$$\Sigma_1 = (a_i, S_j, A_k),$$
$$\Sigma_2 = (e, S_j, A_k),$$
$$\Sigma_3 = (a_i, S_j, e),$$
$$\Sigma_4 = (e, S_j, e).$$

4) *Команда* — это выражение вида

$$\Sigma \to (S_m, x),$$

где Σ — ситуация, S_m — состояние, а x обозначает либо e, либо σ, либо цепочку в словаре V_M.

5) *Абстрактным МП-автоматом* называется конечное множество команд вида

$$\Sigma_l \to (S_{m_l}, x_l);$$

поскольку рассматриваемые автоматы не являются детерминированными, возможны команды с совпадающими левыми частями (т. е. с одинаковыми ситуациями).

Теперь мы можем уточнить, что понимается под процессом вычисления.

Процесс вычисления. Вычисление в МП-автомате задается цепочкой в словаре V_E, записанной на входной ленте и

ограниченной с обеих сторон пробелами:

Перед началом вычислений управляющее устройство находится в начальном состоянии S_0, а входная головка обозревает первый (самый левый) символ данной входной цепочки, при этом в первой (самой левой) ячейке рабочей ленты должен быть записан символ σ, и эта ячейка должна обозреваться рабочей головкой.

Итак, вычисление начинается с физической ситуации (a_{i_1}, S_0, σ) и протекает следующим образом.

Предположим, что МП-автомат попал на каком-то шаге вычисления в ситуацию Σ.

Если в программе автомата нет команды, применимой к Σ, то автомат останавливается.

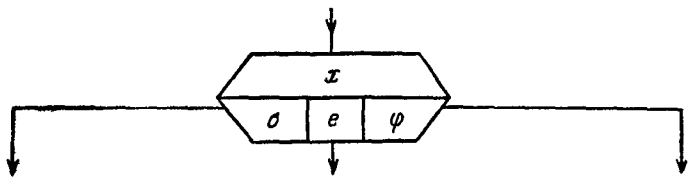
Если такая команда или несколько таких команд имеется, то автомат выбирает из них какую-то одну. Пусть выбрана команда $\Sigma \to (S_m, x)$.

В соответствии с этой командой происходит следующее:

. управляющее устройство переходит в состояние S_m;

.. если первый символ данной ситуации есть e, то входная головка оставляет ленту E неподвижной, а если первый символ — a_i, то головка передвигает ленту так, чтобы обозревать символ, ближайший справа к a_i;

... в зависимости от того, какое значение принимает x, рабочая головка оперирует с лентой M следующим образом:



$\lambda(\sigma) = -1$	$\lambda(e) = 0$	$\lambda(\varphi) = m$
Сдвинуть ленту на одну ячейку вправо, чтобы установить головку против символа, расположенного непосредственно слева от предыдущего символа; стереть предыдущий символ.	Не передвигать ленту; ничего не писать.	Записать последовательность $x = \varphi$ справа от последнего символа; передвинуть ленту на m ячеек влево так, чтобы головка обозревала самый правый из записанных символов.

Автомат, допускающий язык. Будем говорить, что некоторая цепочка во входном словаре *допущена* автоматом, если вычисление, начинающееся с этой цепочки, оканчивается ситуацией

$$(B,\ S_0,\ B).$$

Цепочка называется *допустимой*, если существует вычисление, начинающееся с этой цепочки и заканчивающееся ситуацией (B, S_0, B). Поскольку рассматриваются недетерминированные МП-автоматы, допустимая цепочка может не быть допущена в процессе того или иного конкретного вычисления.

Множество цепочек, допускаемых данным автоматом, представляет собой некоторый язык; говорят, что автомат *допускает* (или *определяет*) этот язык.

Ниже мы изучим класс языков, определяемых классом МП-автоматов; однако сначала приведем два примера.

9.1.3. МП-автомат, определяющий язык L_m

Пусть имеется язык

$$L_m = \{xc\tilde{x} \mid x \in \{a, b\}^*\}.$$

Спрашивается, можно ли построить МП-автомат, определяющий (допускающий) язык L_m.

Наличие центрального маркера c облегчает нашу задачу. Оказывается достаточным копировать заданную цепочку (т. е. переписывать ее на рабочую ленту) до тех пор, пока не появится маркер c. Начиная с этого момента, следует читать копию цепочки x *в обратном направлении*, сравнивая ее буква за буквой с цепочкой, претендующей на роль $\tilde{x}$. Если сравнение оказалось удачным, соответствующая буква стирается. Если предъявленная цепочка правильна, то рабочая лента M в конце концов окажется пустой; в противном случае, когда на M или на E еще остаются некоторые символы, автомат останавливается.

Таким образом, необходимо предусмотреть начальное состояние S_0, состояние чтения S_1 и состояние стирания S_e.

Искомый МП-автомат определяется следующими командами:

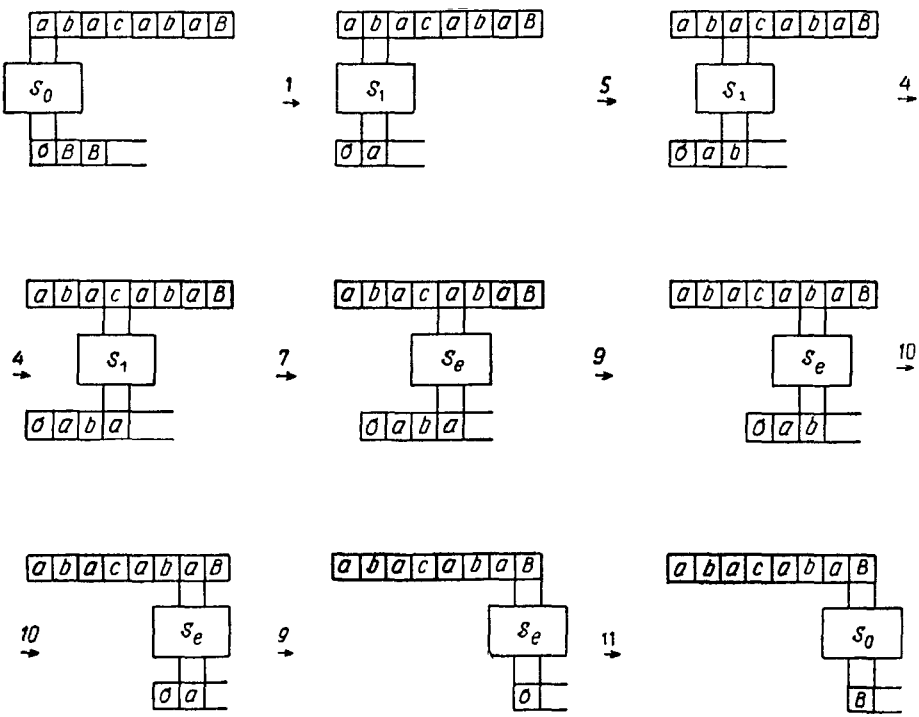
1: $(a,\ S_0,\ \sigma) \to (S_1,\ a)$	первая буква входной цепочки переносится на рабочую ленту M и автомат переходит в состояние S_1 (состояние чтения);
2: $(b,\ S_0,\ \sigma) \to (S_1,\ b)$	
3: $(a,\ S_1,\ a) \to (S_1,\ a)$	буквы входной цепочки одна за другой переносятся на рабочую ленту M, причем автомат остается в состоянии S_1; когда подцепочка x входной цепочки прочтена, лента M содержит x;
4: $(a,\ S_1,\ b) \to (S_1,\ a)$	
5: $(b,\ S_1,\ a) \to (S_1,\ b)$	
6: $(b,\ S_1,\ b) \to (S_1,\ b)$	
7: $(c,\ S_1,\ a) \to (S_e,\ e)$	прочитав центральный маркер c, автомат переходит в состояние S_e (состояние стирания); с рабочей лентой M ничего не происходит;
8: $(c,\ S_1,\ b) \to (S_e,\ e)$	

9: $(a, S_e, a) \to (S_e, \sigma)$ сравнение второй половины входной цепочки $(\tilde{x})$ с записью на ленте M и стирание букв в случае совпадения;

10: $(b, S_e, b) \to (S_e, \sigma)$

11: $(B, S_e, \sigma) \to (S_0, \sigma)$ стирание маркера σ на ленте M.

Пример вычисления. Пусть входная цепочка есть *abacaba*. Тогда последовательность конфигураций нашего автомата такова:



Заметим, что построенный нами МП-автомат является детерминированным: левые части команд попарно различны и не содержат e.

9.1.4. МП-автомат, определяющий «зеркальный» язык без центрального маркера

Рассмотрим теперь язык L_n:

$$L_n = \{x\tilde{x} \mid x \in \{a, b\}^*\},$$

отличающийся от предыдущего только отсутствием центрального маркера (ср. упр. 7.2.6.2). Будем применять ту же самую процедуру обработки цепочек. Поскольку теперь нет маркера, отмечающего середину цепочки, а символы, как и прежде, читаются один за

другим в одном направлении и без возвратов, автомат должен быть недетерминированным [1]). Для каждой позиции необходимо предусмотреть переход в состояние стирания.

Если предъявленная цепочка является неправильной, то она не может быть допущена автоматом. Когда входная цепочка правильна, возможны два случая:

— если переход в состояние стирания произойдет точно в середине цепочки, эта цепочка будет допущена, т. е. ее правильность будет распознана;

— в противном случае автомат остановится или закончит чтение входной цепочки, не очистив рабочую ленту.

Мы предлагаем читателю самостоятельно выписать команды такого МП-автомата.

§ 9.2. АВТОМАТЫ, ПОРОЖДАЮЩИЕ ЯЗЫКИ

До сих пор мы рассматривали

. некоторый реальный механизм (ленты и управляющее устройство);

.. множество команд вида

$$\Sigma \rightarrow (S_m, x),$$

которое задает абстрактный МП-автомат.

Эти команды могут интерпретироваться с помощью указанного механизма, и в результате получается реальный автомат, способный обрабатывать предъявленные ему цепочки: он *допускает* цепочки, принадлежащие рассматриваемому языку, и отвергает все прочие.

Однако множество команд, образующее абстрактный автомат, можно интерпретировать и другим способом: так, чтобы оно определяло реальный автомат иной природы, *порождающий* цепочки того же самого языка.

Порождающий автомат обладает внутренними состояниями, подобными состояниям допускающего автомата (так что мы можем отождествить их); у него имеется такая же лента M, а также лента E, однако на этой последней он не читает, а пишет.

Каждой команде допускающего автомата

$$(a_l, S_j, A_k) \rightarrow (S_m, x)$$

сопоставляется команда порождающего автомата

$$(S_j, A_k) \rightarrow (a_l, S_m, x)$$

(и обратно).

Содержательно команда порождающего автомата означает следующее.

[1]) Это замечание не есть, разумеется, доказательство того, что данный язык не может быть допущен никаким детерминированным МП-автоматом. Однако этот факт может быть строго доказан (см., например, Диковский [1969]). — *Прим. ред.*

Находясь в ситуации (S_j, A_k), порождающий автомат *пишет* a_i на ленте E, затем переходит в состояние S_m и поступает с лентой M в точности так же, как допускающий автомат. Предполагается, что начальным состоянием порождающего автомата является S_0 и что при этом самым левым символом на ленте M всегда является σ.

Поведение порождающего автомата определяется лишь частью ситуации, определяющей поведение допускающего автомата, однако ясно, что всякому правильному вычислению допускающего автомата отвечает правильное вычисление порождающего и наоборот [1]). Порождающий автомат, построенный по допускающему, порождает тот язык, который допускается этим последним. Порождающий автомат выдает правильную цепочку всякий раз, когда, исходя из описанных выше начальных условий, он возвращается в состояние S_0, причем в этот момент лента M пуста.

Пример. Рассмотрим построенный в п. 9.1.3 МП-автомат, допускающий язык $L_m = \{xc\tilde{x}\}$. Вот новая интерпретация команд этого автомата:

1 и 2: записать на лентах E и M один и тот же символ, перейти в состояние S_1;

3, 4, 5 и 6: записать на лентах E и M один и тот же символ, остаться в состоянии S_1;

7 и 8: записать на ленте E центральный маркер и перейти в состояние S_e;

9 и 10: записать на ленте E символ, прочитанный на ленте M, стереть на M этот символ и перейти к чтению следующего символа;

11: дойдя до начального символа σ, стереть его, перейти в состояние S_0 и записать на ленте E пробел.

В качестве упражнения читатель может описать процесс порождения этим автоматом цепочки *abacaba*.

Порождающий автомат может не быть детерминированным даже тогда, когда соответствующий ему допускающий автомат обладает этим свойством. Именно так обстоит дело в только что рассмотренном примере. «Сокращенные» ситуации вроде (S_1, a), (S_1, b) и т. п. допускают применение различных команд, и эта свобода выбора позволяет автомату строить первую половину цепочки как бы наугад; на второй половине цепочки поведение автомата становится детерминированным.

Заключение. Рассмотрим абстрактный набор, состоящий из словарей V_E, V_M и множества команд

$$\{\Sigma_l \rightarrow (S_{m_l}, x_l)\}.$$

---

[1]) Строго говоря, этот факт нуждается в доказательстве; оно, однако, очень просто и может быть предоставлено читателю. — *Прим. ред.*

Можно с равным успехом говорить, что абстрактный автомат, определяемый этим набором, допускает или порождает некоторый язык.

§ 9.3. КЛАСС ЯЗЫКОВ, ДОПУСКАЕМЫХ (ПОРОЖДАЕМЫХ) МП-АВТОМАТАМИ

9.3.1. МП-автомат, допускающий КС-язык

Сейчас мы покажем, что по всякой КС-грамматике можно построить МП-автомат, допускающий (порождающий) язык $L(G)$.

Рассмотрим КС-грамматику G:

$$V_A = \{A_i \mid 1 \leqslant i \leqslant m\}; \quad V_T = \{a_j \mid 1 \leqslant j \leqslant n\};$$

$$\text{аксиома: } S = A_1;$$

$$G: \left| \begin{array}{c} A_i \to \omega_{i,1}, \ \ldots, \ A_i \to \omega_{i,k_i}, \\ 1 \leqslant i \leqslant m \end{array} \right|.$$

Построим по этой грамматике МП-автомат. Его входным словарем $\overline{V}_E$ будет V_T, дополненный символом e; словарем $\overline{V}_M$ — объединение $V_T \cup V_A$, дополненное символами σ и e; его состояниями будут символы [0], [1], [A], ..., [A_m]. Состояние [0] будет начальным. Состояние [1] мы назовем «досинтаксическим», состояния [A_i] — «синтаксическими».

Список команд автомата включает:

1. Начальную команду

$$(e, [0], \sigma) \to ([1], S).$$

2. Команды перехода в синтаксическое состояние

$$(e, [1], A_i) \to ([A_i], \sigma);$$

число этих команд равно числу синтаксических состояний (т. е. числу вспомогательных символов грамматики G).

3. Команды порождения (или допускания)

$$(e, [A_i], e) \to ([1], \omega_{i,k_i}),$$

число которых равно числу правил исходной грамматики.

4. Команды чтения или копирования символов

$$(a_j, [1], a_j) \to ([1], \sigma),$$

число которых равно числу терминальных символов.

5. Заключительную команду

$$(e, [1], \sigma) \to ([0], \sigma).$$

Функционирование автомата при порождении Пусть нашему автомату заданы правильные начальные условия; тогда он переходит в досинтаксическое состояние [1], записывает на рабочей ленте аксиому, запоминает ее, т. е. переходит в соответствующее синтаксическое состояние, и одновременно стирает аксиому.

Находясь в синтаксическом состоянии $[A_i]$, автомат выбирает некоторое правило $A \to \omega_{i,k}$, записывает его правую часть на рабочей ленте, устанавливает рабочую головку так, чтобы она обозревала последний записанный ею символ, и возвращается в досинтаксическое состояние [1].

Вообще, если автомат читает на рабочей ленте M левую часть некоторого правила, то он стирает ее, переходит в соответствующее синтаксическое состояние, записывает на M правую часть того же правила и, вернувшись в досинтаксическое состояние, обозревает последний символ, записанный им на M.

Если же автомат читает на ленте M терминальный символ, то он стирает его, предварительно скопировав на ленте E.

Процесс заканчивается, когда автомат после очередного копирования обозревает символ σ — самый левый символ на ленте M. Ясно, что описанный процесс эквивалентен построению вывода в КС-грамматике с копированием полученных терминальных символов (правда, в обратном порядке).

Функционирование автомата при допускании. Функционирование автомата при допускании (быть может, интуитивно несколько менее ясное) гарантируется «обратимостью» МП-автоматов.

Замечание. Рассматриваемый МП-автомат является недетерминированным. Заметим, однако, что в его командах состояние $[A_i]$ может встретиться в правой части лишь в случае, когда в левой есть A_i. Таким образом, переход в синтаксическое состояние всегда детерминирован.

Пример. Построим описанным только что способом МП-автомат для языка L_m и сравним его с автоматом, рассмотренным выше, в п. 9.1.3.

Пусть имеется КС-грамматика со словарями

$$V_A = \{S\}, \quad V_T = \{a, b, c\}$$

и правилами

$$S \to aSa, \quad S \to bSb, \quad S \to c,$$

Поскольку в этой грамматике имеется ровно один вспомогательный символ — аксиома S, наш автомат будет иметь только одно синтаксическое состояние. Вот программа автомата, соответствующего данной грамматике:

1: $(e, [0], \sigma) \to ([1], S)$

2: $(e, [1], S) \to ([S], \sigma)$

3: $(e, [S], e) \to \begin{cases} ([1], aSa) \\ ([1], bSb) \\ ([1], c) \end{cases}$

4: $\left.\begin{matrix} (a, [1], a) \\ (b, [1], b) \\ (c, [1], c) \end{matrix}\right\} \to ([1], \sigma)$

5: $(e, [1], \sigma) \to ([0], \sigma)$.

Этих команд оказывается меньше, чем команд, приведенных в п. 9.1.3, поскольку последние были детерминированными (они предназначались для допускания цепочек). Читателю будет полезно исследовать, как новые команды связаны со старыми.

9.3.2. Грамматика, порождающая язык, допускаемый МП-автоматом

Построение КС-грамматики, порождающей язык, допускаемый заданным МП-автоматом, также возможно, однако выполняется значительно более сложным способом. В частности, для этого необходима предварительная «нормализация» команд автомата. Здесь мы примем на веру утверждение о возможности такого построения (набросок доказательства дан ниже, в § 15.3). Таким образом, имеет место следующая

Теорема. *Класс языков, допускаемых (порождаемых) МП-автоматами, совпадает с классом КС-языков* (см. ниже, § 15.3).

9.3.3. Упражнения

1. Пользуясь методом п. 9.3.1, построить МП-автомат, допускающий «бесскобочный язык», порождаемый следующей грамматикой:

$$\left|\begin{matrix} S \to CSS \\ S \to \neg S \\ S \to V \\ V \to p \end{matrix}\right|.$$

2. Построить МП-автомат, допускающий язык, порождаемый грамматикой

$$\left|\begin{array}{l} S \to TT \\ T \to Ta \\ T \to bT \\ T \to c \end{array}\right|.$$

3. Язык $L = \{a^p b^q c^r \mid p + q \geqslant r\}$ является КС-языком. Выписать КС-грамматику, порождающую этот язык. Определить, к какому более узкому классу языков он принадлежит. Построить детерминированный МП-автомат, допускающий этот язык.

4. Проблема. Ниже мы рассмотрим некоторые свойства так называемой бесскобочной записи (записи Лукасевича). Для простоты будем обозначать любой бинарный оператор буквой a, а любой операнд — буквой b. Пусть имеется «бесскобочная грамматика» G_p с аксиомой S, терминальными символами a и b и правилами

$$S \to aSS, \quad S \to b.$$

Обозначим через P множество всех терминальных и нетерминальных цепочек, порождаемых грамматикой G_p, и через L — порождаемый ею терминальный язык (таким образом, $L \subset P$).

I. Пусть X — цепочка в алфавите $\{S, a, b\}$; через $\alpha(X)$ обозначается длина X относительно буквы a (т. е. число вхождений a в X), через $\beta(X)$ — длина X относительно букв b и S.

Положим

$$\gamma(X) = \alpha(X) - \beta(X).$$

Например, если $X = ab^3a^2S^4$, то $\alpha(X) = 3$, $\beta(X) = 7$, $\gamma(X) = -4$; если $Y = aba^2b^2ab^3$, то $\alpha(Y) = 4$, $\beta(Y) = 6$, $\gamma(Y) = -2$.

I.1. Доказать импликацию:

$$X \in P \Rightarrow \gamma(X) = -1.$$

I.2. Доказать, что если $X = RQ$, где R и Q не пусты, то

$$X \in P \Rightarrow \gamma(R) \geqslant 0, \quad \gamma(Q) \leqslant -1.$$

I.3. Доказать, что верно и обратное: если $\gamma(X) = -1$ и для любого непустого начала R цепочки X имеет место неравенство $\gamma(R) \geqslant 0$, то $X \in P$.

I.4. Используя результаты упражнений 1—3, построить детерминированный МП-автомат M, допускающий язык L.

I.5. МП-автомат называется *неуправляемым*, если все его команды имеют вид $(a, S_i, e) \to (S_j, x)$. Может ли автомат M быть неуправляемым? Какова была бы — в случае положительного ответа на этот вопрос — роль рабочей ленты?

II. Исследуем теперь структуры, которые G_p сопоставляет цепочкам языка L.

II.1. Объяснить, как можно построить дерево вывода цепочки языка L.

Построить дерево вывода цепочки $a^3bab^2abab^2ab^2$. Доказать, что грамматика G_p однозначна.

II.2. Можно ли построить МП-автомат T, снабженный, кроме входной и рабочей ленты, еще одной лентой — выходной и преобразующей бесскобочную запись в скобочную? Это означает, что любой входной цепочке $f \in L$ автомат T ставит в соответствие (записывая на выходной ленте) цепочку $T(f)$ — «перевод» цепочки f в скобочную запись:

abb переходит в (bab),

$aabbb$ переходит в $((bab)ab)$ и т. п.

III.1. Обозначим через p_i произвольное целое число, большее единицы, и через k — некоторое фиксированное целое положительное число. Доказать, что для любого данного k существует бесконечно много цепочек языка L, имеющих вид

$$a^{p_1}b^{p_1}a^{p_2}b^{p_2} \ldots a^{p_k}b^{p_k}b.$$

III.2. Исходя из этого результата, доказать, что язык L не может быть порожден никакой металинейной грамматикой.

§ 9.4. ПРИЛОЖЕНИЯ КС-ЯЗЫКОВ

Выше было сказано, что КС-языки совпадают с языками, допускаемыми (порождаемыми) автоматами с магазинной памятью; с другой стороны, отмечалось также значение магазинной памяти для программирования. Тем самым устанавливается определенная связь между КС-языками и проблемами программирования для электронных вычислительных машин. Поэтому целесообразно вкратце охарактеризовать приложения КС-языков в программировании и более широко — в области автоматической обработки информации.

9.4.1. Примеры приложений

Важная роль КС-языков объясняется, по всей вероятности, следующими их особенностями:

— в КС-языках возможно неограниченное (циклическое) вставление цепочек друг в друга;

— методы порождения КС-языков весьма естественны с интуитивной точки зрения;

— алгоритмы распознавания КС-языков достаточно просты.

К КС-языкам пришли независимо многие исследователи, работавшие в области формализации синтаксиса:

1) Так называемая нормальная форма Бэкуса

$$\langle A\rangle ::= \langle B\rangle,\ \langle C\rangle \mid n\langle D\rangle$$

представляет собой по существу вариант записи правил КС-грамматики; соответствующая КС-грамматика имеет следующий вид:

$$A \to BC,$$
$$A \to nD,$$

где $A, B, C, D \in V_A$, $n \in V_T$.

2) В лингвистике: модель непосредственно составляющих Р. Уэллза и З. Хэрриса, а также категориальные грамматики И. Бар-Хиллела оказались эквивалентными КС-грамматикам [1]).

3) В автоматическом переводе: почти все грамматики, созданные для целей автоматического перевода, оказались КС-грамматиками.

4) Большинство языков программирования таково, что их синтаксис может быть описан посредством КС-грамматик.

9.4.2. Алгол

Помимо арифметических выражений, в Алголе есть и другие фрагменты, где возможно самовставление. Из них мы укажем здесь два.

а) Возможны выводы вида

⟨непомеченный составной оператор⟩ ⇒

⇒ **(начало)**^{n} ⟨непомеченный составной оператор⟩ **(конец)**^{n}.

Таким образом, алгольная программа, имеющая n символов **начало** в начале и n символов **конец** в конце, считается синтаксически правильной (причем совпадение числа символов **начало** и **конец** существенно для правильности).

б) Возможны условные выражения, где между любыми **если** и **то** могут быть вставлены другие **если** ... **то**. Это соответствует выводам вида

⟨условное выражение⟩ ⇒ **(если** φ_0**)**^{n} ⟨условное выражение⟩ **(**φ_1 **то** φ_2**)**^{n},

где вместо разных вхождений φ_0, φ_1 и φ_2 можно подставлять разные выражения [2]).

---

[1]) О категориальных грамматиках см. Гладкий и Мельчук [1969] стр. 122—123. Доказательство эквивалентности контекстно-свободных и категориальных грамматик см. в следующих работах: Бар-Хиллел, Гайфман и Шамир [1960], Гладкий [1966], стр. 123—132. — *Прим. ред.*

[2]) Ср. предложения на естественном языке вроде *Если о том, что если он сообщит, что если завтра пойдет дождь, то он не приедет, то мы не будем его ждать, он узнает сегодня, то все будет в порядке.* Здесь можно положить $n = 2$; тогда роль (различных вхождений) φ_0 будут играть цепочки *о том, что* и *он сообщит, что*, роль φ_1 — пустая цепочка и *он узнает сегодня*, роль φ_2 — *мы не будем его ждать* и *все будет в порядке*. — *Прим. ред.*

9.4.3. Фортран

Фортран записывается в нормальной форме Бэкуса, т. е. в форме КС-языка. Однако некоторые имеющиеся в Фортране ограничения с помощью нормальной формы Бэкуса выразить не удается.

9.4.4. Лисп (Дж. Маккарти)

Все S-функции, лежащие в основе системы Лисп, могут быть заданы посредством КС-грамматики. Легко убедиться в том, что предлагаемая грамматика соответствует правилам определения. Мы дадим правила порождения S-представлений, являющихся основными списочными структурами Лиспа; при этом мы не учитываем операторы Lambda, Label и Quote, которые тесно связаны с техникой программирования системы.

Предположим, что

$V_T = \{A, \ldots, Z, 0, 1, \ldots, 9, (,), .\}$,

$V_A = \{C_a$ (буква), S_a (атомный символ), S_e (S-выражение), P_r (предикат), E_c (условное выражение), P_a (пара), S_f (S-функция), S_r (S-представление)$\}$.

Правила соответствующей КС-грамматики будут иметь вид

$$S_r \to \left\{\begin{array}{l} S_f \\ P_r \\ S_e \end{array}\right\} \qquad P_r \to \left\{\begin{array}{l} (ATOM,\ S_e) \\ (EQ,\ S_e,\ S_e) \\ E_c \end{array}\right\}$$

$$S_f \to \left\{\begin{array}{l} E_c \\ (CAR,\ S_r) \\ (CDR,\ S_r) \\ (CONS,\ S_r,\ S_r) \end{array}\right. \qquad S_e \to \left\{\begin{array}{l} (S_e,\ S_e) \\ S_a \end{array}\right\}$$

$$S_a \to \left\{\begin{array}{l} (C_a,\ S_a) \\ C_a \end{array}\right\}$$

$$P_a \to \left\{\begin{array}{l} (P_r,\ S_r) \\ (P_r,\ S_r),\ P_a \end{array}\right\}$$

$$E_c \to (COND,\ P_a) \qquad C_a \to \left\{\begin{array}{l} A \\ \vdots \\ Z \\ 0 \\ 1 \\ \vdots \\ 9 \end{array}\right\}.$$

Из самой формы правил ясно, что они обеспечивают самовставление; в частности, они позволяют осуществлять рекурсивное задание списков.

Из теорем о неразрешимости, сформулированных ранее, вытекают определенные практические следствия, уже отмечавшиеся различными авторами.

Неразрешимость проблемы эквивалентности двух грамматик означает, что не существует такой общей процедуры, которая позволяла бы для любых двух данных грамматик решить, порождают ли они одно и то же множество цепочек. Поэтому сравнивать между собой языки или семейства языков, даже таких близких, как варианты Фортрана и Алгола, оказывается нелегко: в каждом конкретном случае приходится искать свои специфические процедуры — большей частью эвристические, позволяющие получить лишь приближенные ответы. То же справедливо и для проблемы неоднозначности данного языка.

Глава X

АВТОМАТНЫЕ ЯЗЫКИ И КОНЕЧНЫЕ АВТОМАТЫ

Языки, о которых пойдет речь в данной главе, были впервые введены и изучены С. К. Клини в связи с исследованием одной простой модели нейрона; эти языки мы будем называть *языками Клини*, или *автоматными языками*[1]) (другие названия: *регулярные языки*, *языки с конечным числом состояний*[2])). В последнее время А-языки нашли широкое применение в исследовании электронных схем, где был получен ряд интересных результатов, относящихся к этим языкам. Совсем в другой связи А-языки были фактически введены в рассмотрение еще А. А. Марковым[3]), который обратил внимание и на их значение для описания естественных языков. Наконец, сравнительно недавно А-языками занялся К. Шеннон. Исходя из тех же вероятностных соображений, что и А. А. Марков, Шеннон исследовал ряд приложений А-языков к изучению естественных языков и пришел к определению некоторого специального класса автоматов, а именно конечных автоматов[4]).

§ 10.1. А-ГРАММАТИКИ

Здесь мы будем рассматривать, полностью абстрагируясь от вероятностных подходов, только те общие свойства А-языков, которые связаны исключительно с «логическими» представлениями. Мы дадим ряд определений (или, скорее, вариантов определения) А-языков, а затем докажем их эквивалентность.

Хотя А-языки являются частным случаем КС-языков, представляется более интересным и полезным с методической точки зрения установить свойства А-языков независимо.

---

[1]) См. выше, стр. 126, в особенности примечание. — *Прим. перев.*

[2]) Термин «языки с конечным числом состояний» (калька с англ. finite-state languages) представляется в качестве русского термина неудачным: языки, понимаемые здесь как множества цепочек, вообще не могут иметь состояний. В английском термине имеется в виду связь с конечным числом состояний «чего-то», а именно автомата, порождающего эти языки — *Прим. перев.*

[3]) Имеется в виду, очевидно, введенное А. А. Марковым (1856—1922) понятие, играющее чрезвычайно важную роль в теории вероятностей и известное в настоящее время под названием «цепи Маркова» — *Прим. ред.*

[4]) А-языки были независимо введены также Н. Хомским. — *Прим. ред.*

10.1.1. Определение

Автоматная грамматика, или *A-грамматика*, имеет алфавит $V = V_A \cup V_T$ (V_A — вспомогательный, V_T — основной), где $V_A \cap V_T = \emptyset$, $V_A = \{A_i \mid 1 \leqslant i \leqslant n\}$, причем A_1 — аксиома, $V_T = \{a_j \mid 0 \leqslant j \leqslant m\}$ и правила вида

$$A_i \to a_j A_k \quad \text{(правосторонние)}$$

или

$$A_i \to A_k a_j \quad \text{(левосторонние)}$$

(в каждой грамматике — **только** правосторонние или **только** левосторонние!), а также вида

$$A_i \to a_j, \quad A_i \to E$$

(словарные или терминальные правила).

Стрелка в $X \to Y$ (как и выше) означает «подставить Y вместо X».

Пример. $V = \{P, Q\} \cup \{a, b\}$; аксиома — P;

$$(G_{\text{прав}}) \ : \ \left| \begin{array}{l} P \to aP \\ P \to aQ \\ Q \to bQ \\ Q \to b \end{array} \right| .$$

Эта грамматика задает язык, состоящий из всевозможных цепочек, устроенных следующим образом: последовательность из a, а за ней — последовательность из b, т. е.

$$\{a^m b^n \mid m, n > 0\}.$$

Тот же самый язык может быть описан другой грамматикой

$$(G_{\text{лев}}) \ : \ \left| \begin{array}{l} P \to Pb \\ P \to Qb \\ Q \to Qa \\ Q \to a \end{array} \right| .$$

Приведем в качестве примера вывод цепочки a^3b^2 в грамматике $(G_{\text{прав}})$:

$$P \to aP \to aaP \to aaaQ \to aaabQ \to aaabb.$$

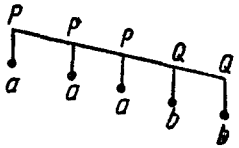
Замечание. Любой конечный язык является А-языком. Множество, содержащее n цепочек, — $\{f_1, f_2, \ldots, f_n\}$, $f_i = a_{i1} \ldots a_{ik_i}$, — может быть описано следующей А-грамматикой:

$$G = \{P \to a_{i1}A_{i1};\ A_{i1} \to a_{i2}A_{i2};\ \ldots;\ A_{i,k_i-2} \to a_{i,k_i-1}A_{i,k_i-1};$$
$$A_{i,k_i-1} \to a_{ik_i} \mid 1 \leqslant i \leqslant n\}.$$

10.1.2. Синтаксические структуры

Подобно КС-грамматике, А-грамматика задает некоторое множество древовидных структур.

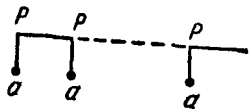
Пример. Цепочке *aaabb*, выведенной в ($G_{\text{прав}}$), отвечает структура



В скобочной форме эта структура имела бы следующий вид:

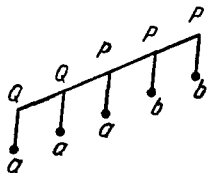
$$\underset{P}{(}a\underset{P}{(}a\underset{P}{(}a\underset{Q}{(}b\underset{Q}{(}b)))))).$$

Структуру вида



иногда называют «правоциклической».

Если бы для вывода цепочки мы воспользовались грамматикой ($G_{\text{лев}}$), то эта цепочка имела бы («левоциклическую») структуру



или в скобочной записи

$$\underset{P}{(}\underset{P}{(}\underset{P}{(}\underset{Q}{(}\underset{Q}{(}a)a)a)b)b).$$

10.1.3. А-языки и Комит

Известный язык программирования Комит[1]) может быть описан посредством А-грамматики. Комит включает команды (правила) следующего вида:

$$\underset{\text{левая часть}}{\Sigma\,\text{SYMB}} = \underset{\text{правая часть}}{\Sigma\,\text{SYMB}'}//\text{INST},\ \ldots,\ \text{NOM}2,$$

где

[1]) См. статью В. Ингве «Язык для программирования задач машинного перевода» в «Кибернетическом сборнике», **6**, 1963, ИЛ, М., стр. 229—266. — *Прим. перев.*

а) NOM 1 — имя (номер) данной команды.

б) Σ SYMB позволяет искать в памяти машины определенные последовательности символов, имеющие в Комите статус единой последовательности.

в) Σ SYMB′ выполняет над символами, обнаруженными левой частью команды, определенные операции.

Таким образом, Σ SYMB содержит операнды, а Σ SYMB′ — операции. Формально Σ SYMB и Σ SYMB′ суть последовательности любых слов, цифр или иных символов, разделенных знаком +. Например:

$$1 + \text{МЯЧ} + \$ + \text{СТАКАН} + 2.$$

Команды Комита выполняются строго слева направо [1]).

г) INST обозначает некоторую последовательность команд, которая должна быть применена к результату работы правой части правила.

д) NOM 2 — имя (номер) команды, которая должна выполняться непосредственно вслед за данной.

Число элементов, фигурирующих в Σ SYMB и в Σ SYMB′, ограничивается только реальным объемом памяти используемой ЭВМ. Однако на соотношения между Σ SYMB и Σ SYMB′ могут быть наложены определенные ограничения. Для того чтобы Комит был A-языком, операции в Σ SYMB′ должны быть определены независимо от операндов. Допустим, однако, что мы хотим, чтобы в ходе трансляции программы, написанной на Комите, можно было проверять, отвечает ли каждой операции в Σ SYMB′ определенный операнд в Σ SYMB; тогда нам придется ввести ограничения, связывающие левую и правую части команды. Если подобные ограничения разрешается неограниченно вставлять друг в друга, то Комит уже не будет A-языком.

§ 10.2. КОНЕЧНЫЕ АВТОМАТЫ

Напомним, что КС-грамматикам мы поставили в соответствие автоматы с магазинной памятью; аналогичным образом мы поставим в соответствие A-грамматикам машины весьма специального вида — так называемые *конечные автоматы.*

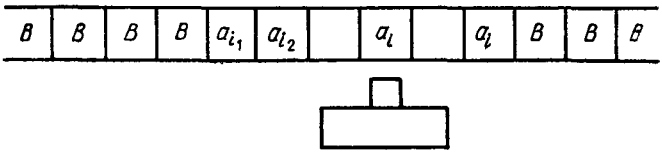
10.2.1. Описание конечных автоматов

Конечный автомат может принимать конечное число *состояний* $S_1, \ldots, S_p$. Он имеет *читающую головку* и *ленту*, которая перемещается под этой головкой. Лента разбита на ячейки; в каждой ячейке может находиться один символ из словаря $V = \{a_i \mid 0 \leqslant i \leqslant n\}$, где $a_0 = B$ играет роль пустого символа.

---

[1]) Именно этим обеспечивается «автоматность» языка. — *Прим ред.*

Одно из состояний автомата — S_0 — выделено и называется *начальным состоянием.*



10.2.2. Функционирование конечного автомата

На содержательном уровне функционирование конечного автомата можно представить себе следующим образом. На ленте записывается цепочка $\alpha \in V^*$; будем считать, что все ячейки слева и справа от α заполнены символами B. Автомат начинает работать в состоянии S_0, причем головка обозревает самый левый символ цепочки α. Прочтя обозреваемый символ, автомат переходит в новое состояние и перемещает ленту на одну ячейку влево. Переход в новое состояние выполняется в соответствии с одной из команд, имеющих вид

$$(a_i,\ S_j) \to S_k.$$

Число команд конечно; левая часть (a_i, S_j) называется *ситуацией автомата*, а правая часть (S_k) есть состояние, в котором автомат будет находиться на следующем шаге (после выполнения данной команды).

Если автомат оказывается в ситуации $(a_{i_1},\ S_{j_1})$, не являющейся левой частью какой-либо команды, то он останавливается. Если после проведения вычислений над цепочкой α автомат возвращается к ситуации (B, S_0), говорят, что автомат *допускает* цепочку α. Множество цепочек, допускаемых данным автоматом, есть по определению *язык, допускаемый этим автоматом.* Класс языков, допускаемых описанным классом, совпадает с классом А-языков.

Пример. Вот множество команд конечного автомата, допускающего язык $\{a^m b^n \mid m, n > 0\}$:

$$(a,\ S_0) \to S_1$$
$$(a,\ S_1) \to S_1$$
$$(b,\ S_1) \to S_2$$
$$(b,\ S_2) \to S_2$$
$$(B,\ S_2) \to S_0.$$

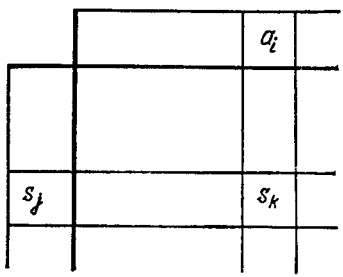
10.2.3. Различные варианты определения конечного автомата

В литературе встречаются различные варианты сформулированного определения. Основные из них связаны со следующими модификациями.

Пустой символ можно использовать в качестве маркера, а можно и вообще отказаться от него.

Вместо того чтобы в конце вычислений возвращаться в начальное состояние S_0, автомат может принимать одно из выделенных *заключительных состояний*; в частности, он может иметь единственное заключительное состояние.

Переходы автомата из одного состояния в другое могут описываться не командами описанного выше вида, а *матрицей переходов*:



или же *функцией переходов*. (Ясно, впрочем, что это различие относится лишь к способу записи.)

Пример. Язык $\{a^m b^n\}$ может быть задан посредством следующей матрицы:

	a	b	$в$
S_0	S_1		
S_1	S_1	S_2	
S_2		S_2	S_0

Функцию переходов мы будем обозначать через $M(a_i, S_j)$. Она определена на подмножестве декартова произведения $V \times S$ и принимает значения из S для случая *детерминированного* автомата, который мы здесь рассматриваем; если же автомат недетерминированный, то значениями функции M являются подмножества S (см. ниже, п. 10.3.1).

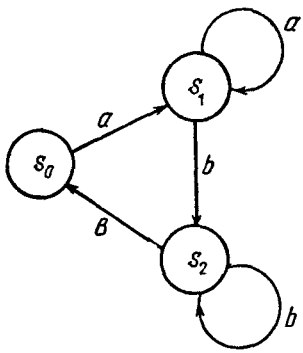
Мы будем доопределять M на множестве $V^* \times S$ следующим образом. Пусть $\varphi \in V^*$ и $\varphi = \varphi_1 a_i$; тогда

$$M(\varphi, S_j) = M(a_i, M(\varphi_1, S_j)).$$

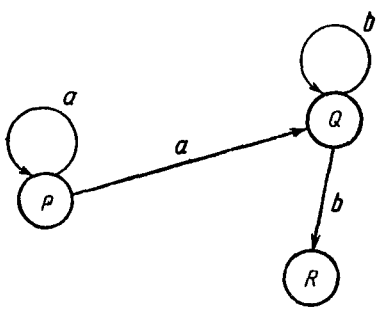
10.2.4. Представление конечного автомата с помощью графа

Произвольному конечному автомату можно поставить в соответствие некоторый «размеченный» граф, действуя следующим образом. Каждому состоянию автомата сопоставляется вершина графа, каждой команде вида $(a_i, S_j) \to S_k$ сопоставляется дуга, ведущая из S_j в S_k и помеченная символом a_i.

Пример. Автомату из предыдущих примеров отвечает следующий граф:



Аналогичным образом можно сопоставить граф А-грамматике: нетерминальным символам ставятся в соответствие вершины графа, каждому правилу вида $A_i \to a_j A_k$ — дуга с пометкой a_j, ведущая из A_i в A_k; каждому правилу вида $A_i \to a_j$ отвечает дуга с пометкой a_j, ведущая из A_i в особую вершину R, не отвечающую никакому нетерминальному символу. Так, для грамматики ($G_{\text{прав}}$) на стр. 160 мы получим следующий граф:



От графа, отвечающего конечному автомату, граф А-грамматики отличается тем, что в первом невозможно, чтобы две дуги, выходящие из одной и той же вершины A и помеченные одним и тем же символом a, входили в разные вершины B и C, тогда как для второго графа такого ограничения нет. Строго говоря, и автомату, и грамматике ставится в соответствие не граф, а мультиграф.

§ 10.3. НЕКОТОРЫЕ ОБОБЩЕНИЯ И ВИДОИЗМЕНЕНИЯ КОНЕЧНЫХ АВТОМАТОВ

10.3.1. Недетерминированные конечные автоматы

Недетерминированный конечный автомат отличается от определенного выше детерминированного (ср. п. 10.2.3) тем, что значениями функции переходов являются не состояния, а множества состояний, или — в терминах команд — возможны различные команды с одинаковыми левыми частями.

В (мульти)графе, отвечающем недетерминированному конечному автомату, возможно, как и в графе А-грамматики, чтобы различные дуги соединяли одну и ту же пару вершин.

Сходство правил А-грамматик и команд недетерминированных конечных автоматов, а также сходство представляющих их графов настолько велико, что эквивалентность А-грамматик и недетерминированных конечных автоматов оказывается почти очевидной. Эта эквивалентность становится совсем очевидной, если принять следующее соглашение: все правила грамматики записываются в виде $A_i \to a_j A_k$, причем терминальные правила имеют вид $A_i \to a_j R$, где R — особый символ, не принадлежащий вспомогательному словарю, A_1 — аксиома.

Грамматике, все правила которой имеют указанный вид, сопоставляется автомат, у которого

— алфавит совпадает с V_T;

— состояния суть символы из $V_A \cup \{R\}$;

— начальное состояние есть A_1, заключительное состояние есть R;

— команды имеют вид $(a_j, A_i) \to A_k$ (для каждого правила грамматики $A_i \to a_j A_k$).

Любая цепочка, порождаемая данной грамматикой, допускается построенным автоматом, и обратно. Совершенно аналогичным образом для любого недетерминированного конечного автомата можно построить эквивалентную ему А-грамматику.

Хотя недетерминированный конечный автомат располагает на первый взгляд «бóльшими возможностями», чем детерминированный, «допускающая сила» обоих типов автоматов оказывается одинаковой. Именно, для любого недетерминированного конечного автомата A с алфавитом V, множеством состояний $S = \{S_0, \ldots, S_p\}$ и функцией переходов M можно построить эквивалентный ему детерминированный конечный автомат A' следующим образом. Состояниями автомата A' будут всевозможные подмножества S; функция переходов M' автомата A' определяется соотношением

$$M(a, \sigma) = \bigcup_{S \in \sigma} M(a, S);$$

начальным состоянием A' будет подмножество S, состоящее из одного состояния — начального состояния автомата A, заключитель-

ными — все подмножества S, содержащие хотя бы одно заключительное состояние A. Детерминированность автомата A' и его эквивалентность автомату A устанавливаются без труда.

10.3.2. Двусторонние автоматы

Еще одно естественное обобщение понятия конечного автомата можно получить, допустив перемещение ленты в обоих направлениях (до сих пор мы считали, что лента движется только в одном направлении). Тогда, обозревая символ в некоторой ячейке ленты, автомат сможет «возвращаться назад», обозревать другой символ, снова протягивать ленту вперед и т. д. вплоть до самого левого символа рабочей цепочки.

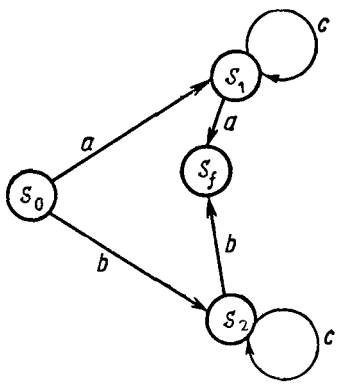
Формально перемещение ленты задается в двустороннем автомате функцией переходов; эта функция имеет такую же область определения, как у обычного конечного автомата, и принимает значения из произведения $S \times \{Л, П\}$, где S — множество состояний, а $Л$ и $П$ обозначают сдвиг ленты на одну ячейку влево и вправо.

Можно доказать, что всякому двустороннему автомату можно поставить в соответствие эквивалентный «обычный», т. е. односторонний, автомат. Таким образом, и это обобщение не изменяет класса допускаемых языков (получаются по-прежнему А-языки).

10.3.3. k-определенные автоматы

Специальный класс конечных автоматов образуют автоматы, у которых текущее состояние зависит только от k последних прочитанных символов рабочей цепочки, причем k для каждого автомата фиксировано. Ответ на вопрос, принадлежит ли предъявленная цепочка языку, порождаемому данным автоматом, можно получить, рассматривая только k последних символов этой цепочки.

Вот пример автомата, не являющегося k-определенным ни при каком k[1]):



[1]) Более того, легко видеть, что ни при каком k не существует k-определенного автомата, эквивалентного данному. — *Прим. ред.*

Понятие k-определенного автомата использовалось для исследования естественных языков, в частности при изучении k-буквенных сочетаний в письменном тексте. Берется матрица, строки которой соответствуют всем последовательностям из k букв рассматриваемого языка, а столбцы — буквам этого языка. На пересечении строки i и столбца j записывается вероятность того, что буква a_j встретится в тексте вслед за последовательностью букв p_i. Порождение слова k-определенным автоматом, представленным посредством подобной матрицы, выполняется так. Находясь в состоянии, которое соответствует самой левой в слове последовательности из k букв: $a_{i_1}a_{i_2} \ldots a_{i_k}$, автомат выдает в зависимости от этих k букв букву a_j и переходит в состояние, соответствующее последовательности $a_{i_2}a_{i_3} \ldots a_{i_k}a_j$.

§ 10.4. СВОЙСТВА ЗАМКНУТОСТИ. ПРЕДСТАВЛЕНИЕ А-ЯЗЫКОВ ПО КЛИНИ

Покажем, что А-языки образуют булеву алгебру.

10.4.1. Объединение

Объединение А-языков есть А-язык. Это утверждение доказывается так же, как аналогичное утверждение для КС-языков.

10.4.2. Дополнение

Рассмотрим (детерминированный) конечный автомат A, допускающий язык L. Построим по A автомат A' следующим образом (нам будет удобно говорить при этом не о самих автоматах, а о соответствующих графах). Все вершины и дуги графа автомата A войдут в граф A'. Кроме того, граф A' будет содержать еще одну вершину R, в которую будут вести следующие дуги: а) если из какой-либо вершины S исходного графа не выходит ни одной дуги, помеченной символом a, то из S в R идет дуга, помеченная этим символом; б) для каждого основного символа a из R в R идет дуга («петля»), помеченная этим символом. Начальная вершина нового графа будет совпадать с начальной вершиной старого; заключительными вершинами будут R и все незаключительные вершины старого графа. Из детерминированности A сразу следует, что A' не допустит ни одной цепочки языка L. В то же время любая цепочка в основном словаре, не принадлежащая L, будет допущена автоматом A': если она является началом какой-либо цепочки из L, она будет «написана» на некотором пути, ведущем из начальной вершины в незаключительную вершину старого графа, в противном случае — на пути, ведущем из начальной вершины в R. Таким образом, A' допускает дополнение L.

Итак, *дополнение А-языка есть А-язык.*

Из этого результата вместе с предыдущим вытекает, что и *пересечение двух А-языков есть А-язык.*

10.4.3. Произведение и итерация

Произведение А-языков есть А-язык. Для доказательства достаточно рассмотреть А-грамматики G_1 и G_2 с непересекающимися вспомогательными словарями и объединить множества их правил, заменив при этом каждое правило вида $A \to a$ грамматики G_1 правилом $A \to aB_0$, где B_0 — аксиома грамматики G_2.

Итерация А-языка есть А-язык. Действительно, если G есть А-грамматика с аксиомой A_0, то добавление к ней всевозможных правил вида $A \to aA_0$, где $A \to a$ — правило G, дает А-грамматику, порождающую $L(G)^*$. Легко видеть также, что *обращение А-языка есть А-язык.*

10.4.4. Теорема Клини

Класс А-языков совпадает с наименьшим классом языков, содержащим все конечные языки и замкнутым относительно операций объединения, умножения и итерации.

Доказательство. В одну сторону теорема следует из результатов п. 10.4.1—10.4.3, поскольку все конечные языки являются А-языками. Докажем обратное утверждение, т. е. покажем, что всякий А-язык может быть получен из конечных языков с помощью трех перечисленных операций. Для языка, порождаемого А-грамматикой, граф которой не содержит ни одной дуги, это утверждение очевидно. Пусть оно доказано для А-грамматик, графы которых содержат не более k дуг, и пусть G есть А-грамматика с k дугами в графе. Выбросим из этого графа некоторую дугу, ведущую из начальной вершины A_0 в вершину B и помеченную символом a. Грамматику с таким графом обозначим G'. Грамматики, полученные из G': (1) переносом начальной вершины в B; (2) объявлением A_0 единственной заключительной вершиной; (3) совместным выполнением предыдущих двух преобразований, — обозначим через G_1, G_2 и G_3 соответственно. Нетрудно видеть, что

$$L(G) = L(G') \cup L(G_2)(aL(G_3))^* aL(G_1).$$

10.4.5. Разрешимые алгоритмические проблемы

Применительно к конечным автоматам могут быть сформулированы многие из тех проблем, которые возникали для машин Тьюринга и автоматов с магазинной памятью. Здесь эти проблемы оказываются, как правило, разрешимыми. А именно, существуют алгоритмы, позволяющие получить ответ «да» или «нет» на следующие вопросы:

— допускает ли данный конечный автомат пустой язык;
— допускает ли данный автомат бесконечный язык;
— являются ли два данных конечных автомата эквивалентными.

10.4.6. Представление автоматных языков по Клини

Теорема Клини позволяет дать следующее простое рекурсивное определение А-языков.

Пусть имеется словарь V. Прежде всего мы определим *представляющие выражения* для цепочек из V^*:

1) цепочка из V^* есть представляющее выражение;
2) если X_1 и X_2 — представляющие выражения, то X_1X_2 — тоже представляющее выражение,
3) если X_i $(1 \leqslant i \leqslant n)$ — представляющие выражения, то $(X_{i_1}, \ldots, X_{i_m})^*$, где $i_k = \{1, \ldots, n\}$, — тоже представляющие выражения;
4) если $X_1, \ldots, X_n$ — представляющие выражения, то $X_1 \cup \ldots \ldots \cup X_n$ — тоже представляющее выражение;
5) других представляющих выражений нет.

Эти выражения *представляют* множества цепочек из V^* (языки в V) следующим образом:

1) цепочка из V^* представляет самое себя (точнее, соответствующий «одноцепочечный язык»);
2) если X_1 представляет множество Σ_1, а X_2 — множество Σ_2, то X_1X_2 представляет $\Sigma_1\Sigma_2$, т. е. множество всех цепочек $\varphi_1\varphi_2$, таких, что $\varphi_1 \in \Sigma_1$ и $\varphi_2 \in \Sigma_2$;
3)—4) если X_i $(1 \leqslant i \leqslant n)$ представляют множества цепочек Σ_i $(1 \leqslant i \leqslant n)$, то $(X_{i_1}, \ldots, X_{i_m})^*$ представляет множество $\{\Sigma_{i_1} \cup \ldots \ldots \cup \Sigma_{i_m}\}^*$, а $X_1 \cup \ldots \cup X_n$ — множество $\Sigma_1 \cup \ldots \cup \Sigma_n$.

В силу теоремы Клини всякий А-язык может быть задан некоторым представляющим выражением.

Примеры.

ab представляет язык, содержащий ровно одну цепочку ab;

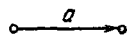
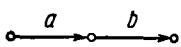
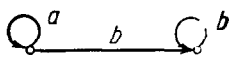
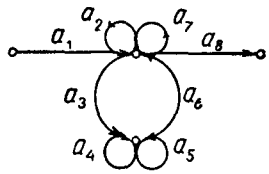
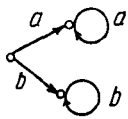
$(a)^*$ представляет язык $\{a^n \mid n \geqslant 0\}$;

$(a)^*(b)^*$ представляет язык $\{a^p b^q \mid p, q \geqslant 0\}$;

$(a, b)^*$ представляет свободную полугруппу с образующими $\{a, b\}$;

$(a)^* \cup (b)^*$ представляет язык $\{a^p \mid p \geqslant 0\} \cup \{b^q \mid q \geqslant 0\}$.

Представляющим выражениям можно ставить в соответствие графы; например:

	Представляющее выражение	Соответствующий граф
1)	a	
2)	ab	
3)	$(a)^*$	
4)	$(a)^*(b)^*$	
5)	$a_1(a_2, a_3(a_4, a_5)^* a_6, a_7)^* a_8$	
6)	$(a^*) \cup (b)^*$	

Для четвертого представляющего выражения обе вершины заключительные, а для шестого выражения все три вершины заключительные. Эти графы совпадают, очевидно, с графами соответствующих А-грамматик.

§ 10.5. А-ЯЗЫКИ И КС-ЯЗЫКИ

10.5.1. Самовставление

Особенность А-языков по сравнению с КС-языками заключается в том, что всякий А-язык может быть порожден грамматикой, в которой нет ни одного вспомогательного символа с самовставлением, тогда как существуют КС-языки, которые не порождаются никакими КС-грамматиками без самовставления.

Рассмотрим язык $L = \{a^n b^n \mid n > 0\}$, порождаемый КС-грамматикой:

$$\left| \begin{array}{l} S \to aSb \\ S \to ab \end{array} \right|.$$

Докажем, что он не может быть порожден никакой А-грамматикой. Допустим, что А-грамматика, порождающая язык L, существует. Поскольку L бесконечен, в этой грамматике должны быть возможными выводы вида $A_i \Rightarrow xA_i$, где x имеет вид a^p; следовательно, в этой грамматике должны быть возможными и выводы вида

$$A_i \Rightarrow yA_j, \qquad y = a^q b^r \mid q,\ r \geqslant 0,$$
$$A_j \Rightarrow zA_j, \qquad z = b^m.$$

Тогда данная грамматика будет порождать некоторые цепочки вида $a^p b^q$, где $p \neq q$ [1]).

А-язык может быть порожден и грамматикой с самовставлением. Однако на вопрос, можно ли по данной КС-грамматике определить, порождает она А-язык или нет, приходится ответить отрицательно; это следует из нераспознаваемости автоматности дополнения к КС-языку (см. п. 8.1.2) [2]).

§ 10.6. ОДНОСТОРОННИЕ КОНЕЧНЫЕ ПРЕОБРАЗОВАТЕЛИ

10.6.1. Автомат типа $\mathfrak{T}$

Пусть $\mathfrak{A}$ — конечный автомат; снабдим его выходной лентой, на которой он будет писать после каждого такта работы некоторый символ из *выходного алфавита* $V_S = \{b_i\}$. У нас получится автомат нового типа $\mathfrak{T}$, принадлежащий к классу *односторонних конечных преобразователей*. Команды автомата $\mathfrak{T}$ имеют вид

$$(a_i,\ S_j) \rightarrow (S_k,\ b_l);$$

когда преобразователь $\mathfrak{T}$ посредством конечного автомата $\mathfrak{A}$, входящего в его состав, определяет допустимость некоторой предъявленной цепочки f, он одновременно дает «перевод», или *транскрипцию*, этой цепочки в $\{b_i\}^*$. Транскрипция цепочки f, полученная преобразователем $\mathfrak{T}$, — *транскрипция* f *относительно* $\mathfrak{T}$, — обозначается через $\mathfrak{T}(f)$.

Автомат $\mathfrak{T}$ можно изобразить с помощью диаграммы (графа), вершины которой соответствуют его состояниям. Команде $(a_i, S_j) \rightarrow (S_k, b_l)$ отвечает в этой диаграмме дуга, помеченная парой (a_i, b_l) и ведущая из S_j в S_k. Легко видеть, что множество транскрипций цепочек, допускаемых преобразователем $\mathfrak{T}$, есть А-язык.

Если в командах преобразователя $\mathfrak{T}$ поменять ролями a_i и b_l, т. е. перейти к командам вида

$$(b_l,\ S_j) \rightarrow (S_k,\ a_i),$$

---

[1]) Можно доказать, что любая КС-грамматика, порождающая неавтоматный язык, имеет самовставление. — *Прим. ред.*

[2]) Здесь используется то обстоятельство, что, в силу п. 10.4.2, язык тогда и только тогда является А-языком, когда таковым является его дополнение. — *Прим. ред.*

мы получим преобразователь $\mathfrak{T}^{-1}$, обратный к $\mathfrak{T}$. Обратный преобразователь изображается диаграммой, которую легко построить по предыдущей, пометив дуги парами (b, a) вместо (a, b).

10.6.2. Формальное определение

В наиболее общем виде односторонний конечный преобразователь $\mathfrak{T}$ определяется заданием следующих объектов:

1) входной алфавит $V_E = \{a_1, \ldots\}$ и выходной алфавит $V_S = \{b_1, \ldots\}$, каждый из которых содержит по меньшей мере две буквы;

2) конечное множество состояний $\Sigma = \{S_0, \ldots\}$;

3) отображение, определяющее переход в новое состояние (функция переходов):

$$\Sigma \times V_E \to \Sigma;$$

4) выходное отображение:

$$\Sigma \times V_E \to V_S^*.$$

Преобразователь, приведенный в начальное состояние S_0, читает самый левый символ цепочки $f = a_{i_1} a_{i_2} \ldots a_{i_n}$; затем он переходит в новое состояние, отвечающее паре (S_0, a_{i_1}), и пишет на выходной ленте символ (или цепочку), отвечающий этой же паре. Затем он читает очередной символ цепочки f, т. е. a_{i_2}, и т. д.

Описанные преобразователи называются односторонними потому, что транскрипция выполняется в ходе чтения предъявленной цепочки строго в одном направлении — слева направо.

Если преобразователь оказывается в ситуации, для которой его программа не содержит соответствующей команды, он останавливается. Если же он доходит до конца читаемой цепочки, то одновременно он заканчивает и ее транскрипцию.

10.6.3. Транскрипция КС- языка

Рассмотрим теперь КС-грамматику G и односторонний конечный преобразователь $\mathfrak{T}$. Спрашивается, к какому классу принадлежит язык $\mathfrak{T}(L)$, представляющий собой транскрипцию языка $L = L(G)$ (точнее, пересечения $L(G)$ с языком, который будет допускать преобразователь $\mathfrak{T}$, если лишить его выхода) относительно $\mathfrak{T}$.

Можно показать, что $\mathfrak{T}(L)$ есть КС-язык; мы построим для него КС-грамматику G'.

Пусть $V_A = \{A_i \mid 0 \leqslant i \leqslant n\}$ — вспомогательный словарь грамматики G, где A_0 — аксиома; $V_T = \{a_j\}$; $\{S_k\}$ — множество состояний преобразователя T; S_0 — его начальное состояние, а $\{S_f\}$ — его заключительные состояния.

Правила грамматики G' определяются следующим образом.

Аксиомой в G' является новый символ A_0'; прочие нетерминальные символы — это тройки вида (S_i, α, S_j), где S_i и S_j суть

состояния преобразователя $\mathfrak{T}$, а α — символ (терминальный или вспомогательный), используемый грамматикой G. Кроме того,

1) для всякого заключительного состояния S_f грамматики G выражение $A'_0 \to (S_0, A_0, S_f)$ есть правило грамматики G';

2) если $A_m \to \alpha_1 \ldots \alpha_h$ есть правило грамматики G, то для любых $i, j, \beta_1, \ldots, \beta_{h-1}$ грамматика G' содержит правило

$$(S_i, A_m, S_j) \to (S_i, \alpha_1, S_{\beta_1})(S_{\beta_1}, \alpha_2, S_{\beta_2}) \ldots (S_{\beta_{h-1}}, \alpha_h, S_j);$$

3) если $(a_j, S_p) \to (S_q, b_l)$ — команда преобразователя T, то G' содержит правило

$$(S_p, a_j, S_q) \to b_l.$$

Нетрудно показать, что $L(G') = \mathfrak{T}(L)$.

Для частного случая, когда преобразователь просто копирует входную цепочку, т. е. когда его команды имеют вид $(a_i, S_j) \to (S_k, a_i)$, из доказанного утверждения получается следующая

Теорема. *Пересечение КС-языка и А-языка есть КС-язык.*

Глава XI

ЗАДАНИЕ ЯЗЫКОВ С ПОМОЩЬЮ СИСТЕМ УРАВНЕНИЙ

§ 11.1. ФУНКЦИИ, АРГУМЕНТАМИ И ЗНАЧЕНИЯМИ КОТОРЫХ ЯВЛЯЮТСЯ ЯЗЫКИ

11.1.0. Операции

Выше мы определили ряд операций над языками, из которых здесь нам понадобятся две:

. теоретико-множественное объединение $L \cup M$; эта операция коммутативна и ассоциативна;

.. умножение LM, которое заключается в образовании декартова произведения двух языков с последующей конкатенацией членов каждой пары:

$$LM = \{xy \mid x \in L,\ y \in M\}.$$

Умножение языков ассоциативно, но не коммутативно, как и операция конкатенации цепочек, к которой оно восходит.

Заметим, что умножение языков дистрибутивно относительно объединения:

$$L(L_1 \cup L_2) = LL_1 \cup LL_2; \qquad (L_1 \cup L_2)L = L_1L \cup L_2L.$$

Мы можем теперь ввести понятия переменной, формального выражения и функции.

11.1.1. Переменные и выражения

Рассмотрим семейство языков $\{L_1, L_2, \ldots\}$; будем считать, что они определены над одним и тем же терминальным словарем V_T и что мы имеем вспомогательный словарь V_A, который годится для описания грамматик всех этих языков.

Под языком мы будем понимать, как и раньше, всякое множество терминальных цепочек.

«Языковые» переменные. Нас будут интересовать переменные $\mathfrak{L}, \mathfrak{M}, \mathfrak{N}, \ldots$, значениями которых служат языки.

Пример. Пусть рассматривается класс КС-языков и введена переменная $\mathfrak{L}$, пробегающая этот класс. Одним из «допустимых» значений переменной $\mathfrak{L}$ является язык L_m:

$$\mathfrak{L} = L_m = \{xc\tilde{x} \mid x \in \{a, b\}^*\}.$$

Формальные выражения. Используя *константы* $t, a, \ldots$ (фиксированные терминальные цепочки), переменные $\mathfrak{L}, \mathfrak{M}, \mathfrak{N}, \ldots$ и символы операций, мы можем строить формальные выражения вроде

$$t\mathfrak{L}; \quad t\mathfrak{L} \cup a; \quad \mathfrak{L}a\mathfrak{M} \cup a\mathfrak{L}t; \ldots.$$

Если значениями переменных, входящих в подобное выражение, являются те или иные языки, то и значением самого выражения будет некоторый язык.

Примеры. 1) Пусть имеется алфавит $\{а, и, к, л, о, п, р, т, ч, ь, я\}$ и константа $t = $ '*про*'; рассмотрим выражение $t\mathfrak{L}$. Придадим переменной $\mathfrak{L}$ значение L, где

$$L = \{кричать,\ лаять,\ к\};$$

L — конечный язык, состоящий из трех терминальных цепочек.

Соответствующим значением выражения $t\mathfrak{L}$ является

$$L' = \{прокричать,\ пролаять,\ прок\},$$

т. е. другой конечный язык, состоящий также из трех цепочек.

2) Придадим константе t значение ab и переменной $\mathfrak{L}$ — значение L_m, где

$$L_m = \{xc\tilde{x} \mid x \in \{a, b\}^*\}.$$

Тогда выражение $t\mathfrak{L}$ примет значение

$$\{abc,\ abaca,\ abbcb,\ ababcba, \ldots\}$$

— язык, который получается приписыванием цепочки ab слева ко всем цепочкам языка L_m.

Многочлены. Если формальное выражение образовано из терминальных констант и языковых переменных с помощью только двух операций — умножения и объединения, — мы будем называть его *многочленом*; порядок выполнения операций объединения («порядок членов») в многочлене безразличен.

Пример. Выражение $ab\mathfrak{L} \cup a\mathfrak{M} \cup \mathfrak{M}ba\mathfrak{L}$ есть многочлен от двух переменных $\mathfrak{L}$ и $\mathfrak{M}$, состоящий из трех «членов».

11.1.2. Функции и уравнения

Ясно, что задание формального выражения равносильно заданию некоторой функции. Если каждой переменной в выражении отвечает некоторое семейство языков, то соответствующие значения этого выражения также образуют некоторое вполне определенное семейство языков. Таким образом может быть задано отображение множества допустимых наборов значений аргументов на множество допустимых значений выражения, т. е. функция, определенная на семействе наборов языков и в качестве значений принимающая языки.

Подчеркнем тот факт, что по крайней мере здесь, когда мы рассматриваем в качестве значения «языковой» переменной $\mathfrak{L}$ язык L, сам L рассматривается сугубо с теоретико-множественной точки зрения; в частности, цепочки языка L не снабжены никакими характеристиками (т. е. им не приписаны синтаксические структуры).

Пример. Рассмотрим в качестве области определения переменной $\mathfrak{M}$ класс линейных языков в алфавите $\{a, b, c\}$. Тогда выражение

$$b\,\mathfrak{M}\,b \cup \mathfrak{M}\,a\,\mathfrak{M}$$

задает функцию, которая имеет эту область определения и значениями которой являются языки.

Уравнения. Для обозначения только что введенных функций мы будем использовать классические обозначения, а именно

$$y = f(\mathfrak{L}, \mathfrak{M}, \ldots).$$

Если считать значения некоторых из переменных $\mathfrak{L}, \mathfrak{M}, \ldots$ известными, а других — неизвестными, мы получим *уравнения*, в которых неизвестные являются языками; возникает вопрос о методах решения таких уравнений.

Обратно, если дан некоторый язык, можно пытаться построить интересные уравнения, для которых данный язык будет решением.

Сначала мы изучим эти новые понятия на примере одного частного случая.

11.1.3. Уравнение, связанное с «зеркальным языком» L_m

Пусть имеется язык L_m в алфавите $\{a, b, c\}$, порождаемый следующей грамматикой:

$$\left|\begin{array}{l} S \to aSa \\ S \to bSb \\ S \to c \end{array}\right|.$$

Рассмотрим функцию, заданную выражением $a\mathfrak{L}a$.

Если значением переменной $\mathfrak{L}$ является L_m, то эта функция принимает значение $L_m' = aL_ma$; язык L_m' есть подъязык языка L_m, а именно — он содержит все те и только те цепочки языка L_m, которые начинаются и оканчиваются символом a.

Рассмотрим, далее, выражение $a\mathfrak{L}a \cup b\mathfrak{L}b$. При $\mathfrak{L} = L_m$ задаваемая им функция принимает в качестве значения язык, получаемый из L_m выбрасыванием цепочки c.

Итак, $L_m = aL_ma \cup bL_mb \cup c$.

Значит, язык L_m является *одним* из решений уравнения

$$\mathfrak{L} = a\mathfrak{L}a \cup b\mathfrak{L}b \cup c. \qquad (*)$$

11.1.4. Решение полученного уравнения

Можно задать вопрос, является ли язык L_m (в алфавите $\{a, b, c\}$) единственным решением только что выписанного уравнения. Запишем это уравнение в виде

$$\mathfrak{L} = f(\mathfrak{L})$$

и попытаемся решить его.

Поскольку значение третьего члена правой части уравнения (*) известно, очевидно, что всякое решение L должно содержать однобуквенную цепочку c, т. е. содержать язык $L_1 = \{c\}$. Поэтому всякое решение обязательно содержит также и язык

$$L_2 = aL_1a \cup bL_1b \cup c = \{aca,\ bcb,\ c\}.$$

Однако всякое решение нашего уравнения, поскольку оно содержит L_2, будет содержать также

$$\begin{aligned} L_3 &= aL_2a \cup bL_2b \cup c = \\ &= a\{aca,\ bcb,\ c\}a \cup b\{aca,\ bcb,\ c\}b \cup c = \\ &= \{aacaa,\ abcba,\ bacab,\ bbcbb,\ aca,\ bcb,\ c\}. \end{aligned}$$

Вообще, решение L обязательно должно содержать множества

$$L_1 = \{c\},\ L_2 = f(L_1),\ \ldots,\ L_n = f(L_{n-1}),\ \ldots,$$

которые получаются посредством рекуррентного процесса, причем каждое из них содержит в себе предыдущее. Следовательно, L содержит объединение всех этих множеств — мы обозначим его $\lim\limits_{n\to\infty} f(L_n)$. Но это объединение есть не что иное, как «зеркальный язык» L_m. Итак, всякое решение нашего уравнения содержит L_m, т. е. L_m оказывается *наименьшим* решением нашего уравнения. Мы будем называть наименьшее решение просто *решением* [1]).

Вариант записи. При решении уравнения $\mathfrak{L} = f(\mathfrak{L})$ мы использовали язык $L_1 = \{c\}$, однако константу мы могли бы получить и иным путем, а именно введя в рассмотрение «идеальный элемент» ω [2]), являющийся нулем относительно конкатенации (т. е. такой, что $a\omega = \omega a = e$, где e — пустая цепочка). Положив

$$L_0 = \omega,$$

получим

$$f(L_0) = \{a\omega a \cup b\omega b \cup c\} = c = L_1,$$

что унифицирует запись рекуррентного процесса.

11.1.5. Индуцированная структура

В начале этой главы мы напоминали, что умножение языков и конкатенация цепочек — ассоциативные операции. Если даны три языка L_1, L_2, L_3, то мы имеем

$$(L_1L_2)L_3 = L_1(L_2L_3),$$

---

[1]) Легко видеть, что это решение — единственное. В самом деле, если L' — какое-либо решение данного уравнения и $x \in L'$, то либо x есть c, либо x имеет вид aya или byb, где $y \in L'$; однако то же верно и относительно y и т. д., так что x должна иметь вид $zc\tilde{z}$ ($z \in \{a, b\}^*$). Таким образом, $L' \subset L_m$, а поскольку L_m — наименьшее решение, то $L' = L_m$. — *Прим. ред.*

[2]) «Идеальный элемент» ω не следует смешивать с пустой цепочкой, которая является единицей относительно конкатенации.

что позволяет не писать скобок. Однако сохранение скобок (или заменяющих их символов) обеспечивает представление в явном виде «истории» получаемых цепочек.

Пусть имеются языки L_1, L_2, L_3, заданные перечислением их цепочек:

$$L_1 = \{x_1, x_2, \ldots, x_i, \ldots\},$$
$$L_2 = \{y_1, y_2, \ldots, y_j, \ldots\},$$
$$L_3 = \{z_1, z_2, \ldots, z_k, \ldots\}.$$

Образуем произведение L_1L_2, а затем $(L_1L_2)L_3$; тогда результирующий язык будет состоять из цепочек $x_iy_jz_k$, имеющих структуру $((x_iy_j)z_k)$. Если же образовать сначала произведение L_2L_3, а затем $L_1(L_2L_3)$, то полученный язык будет содержать те же самые цепочки, но с другой структурой: $(x_i(y_jz_k))$. Точно так же язык aL_1 состоит из цепочек ax_i со структурой $(a(x_i))$ и т. п.

Потребуем, чтобы при решении уравнений типа $\mathfrak{L} = f(\mathfrak{L})$ сохранялась структура цепочек, индуцированная самим ходом решения.

Пример решения с сохранением структуры. Решим указанным образом то же уравнение $\mathfrak{L} = f(\mathfrak{L})$:

$\mathfrak{L}_0 = \omega$,

$\mathfrak{L}_1 = \{a\omega a \cup b\omega b \cup c\} = \{(c)\}$,

$\mathfrak{L}_2 = \{a\,(c)\,a \cup b\,(c)\,b \cup c\} = \{(a\,(c)\,a),\ (b\,(c)\,b),\ (c)\}$,

$\mathfrak{L}_3 = \{a\,\{(a\,(c)\,a),\ (b\,(c)\,b),\ (c)\}\,a,\ b\,\{(a\,(c)\,a),\ (b\,(c)\,b),\ c\}\,b,\ (c)\} =$

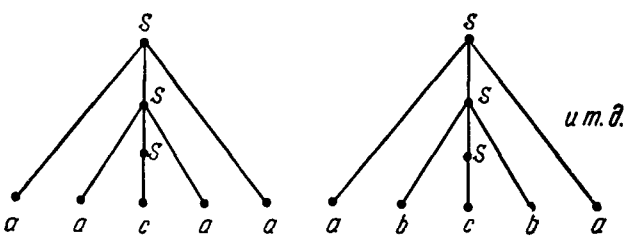
$= \{(a\,(a\,(c)\,a)\,a),\ (a\,(b\,(c)\,b)\,a),\ (b\,(a\,(c)\,a)\,b),\ (b\,(b\,(c)\,b)\,b),\ (a\,(c)\,a),$
$(b\,(c)\,b),\ (c)\}$ и т. д.

Тем самым мы снабдим структурами все цепочки языка L_m.

Эти структуры идентичны тем структурам, которые сопоставляет цепочкам языка L_m порождающая его КС-грамматика с правилами

$$S \to aSa,\ S \to bSb,\ S \to c.$$

Так, для L_3 получаются следующие структуры:



Отсутствие пометок при скобках в нашем примере связано с тем фактом, что соответствующая грамматика имеет лишь один вспомогательный символ.

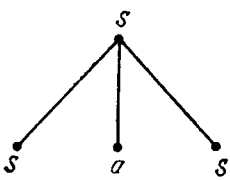
Ясно, однако, что приписывание цепочкам структур наиболее интересно в случае неоднозначных грамматик. Предыдущая грамматика была однозначной, поэтому теперь мы рассмотрим другую грамматику, порождающую «бинарные формулы», т. е. цепочки, получаемые с помощью некоторого бинарного оператора.

11.1.6. Пример для случая неоднозначной грамматики

Рассмотрим следующую КС-грамматику:

$$V_A = \{S\}, \quad V_T = \{a, b\}, \quad | S \to SaS, \ S \to b |.$$

Любой вывод, за исключением вывода $S \to b$, начинается так:



и каждое из полученных S может быть началом вывода цепочки, принадлежащей порождаемому данной грамматикой языку. Этому языку отвечает уравнение

$$\mathfrak{L} = \mathfrak{L}a\mathfrak{L} \cup b,$$

которое мы сейчас решим, сохраняя в цепочках структуру, индуцируемую ходом решения.

Итак,

$\mathfrak{L}_0 = \omega,$

$\mathfrak{L}_1 = \{\omega a \omega \cup b\} = \{(b)\},$

$\mathfrak{L}_2 = \{\{(b)\}\, a\, \{(b)\} \cup b\} = \{((b)\, a\, (b)),\ (b)\},$

$\mathfrak{L}_3 = \{\{((b)\, a\, (b)),\ (b)\}\, a\, \{((b)\, a\, (b)),\ (b)\} \cup b\} =$

$= \{((((b)\, a\, (b))\, a\, ((b)\, a\, (b))),\quad (((b)\, a\, (b))\, a\, (b)),\quad ((b)\, a\, ((b)\, a\, (b))),$

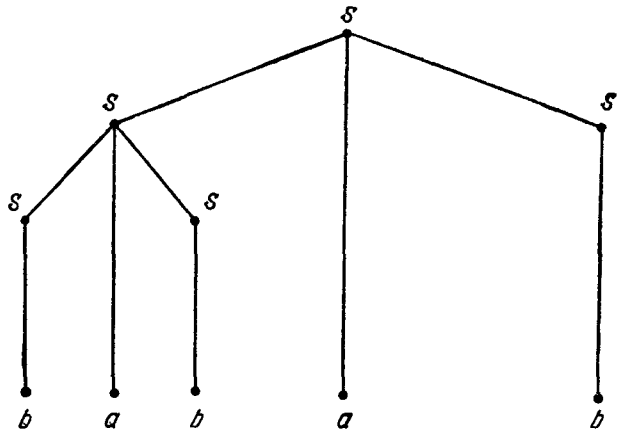
$((b)\, a\, (b)),\ (b)\}$ и т. д.

Сняв все скобки, мы получим

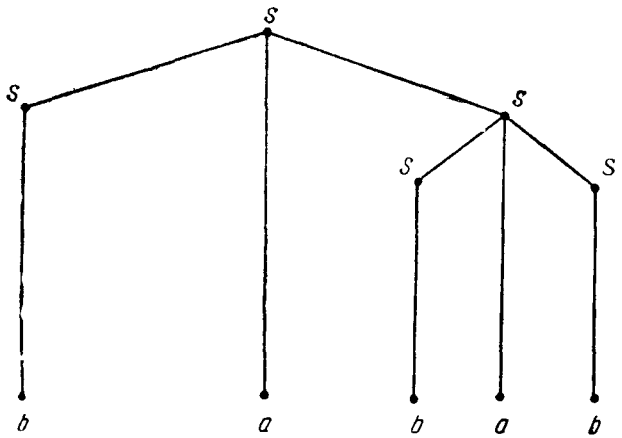
$$\mathfrak{L}_3 = \{bababab,\ babab,\ babab,\ bab,\ b\}.$$

Мы видим, что цепочка *babab* была получена дважды. С теоретико-множественной точки зрения достаточно иметь один ее экзем-

пляр; однако разные экземпляры имеют разную структуру. Первый отвечает выводу



и имеет структуру $(((b)a(b))a(b))$, а второй получается в результате другого вывода, «симметричного» первому:



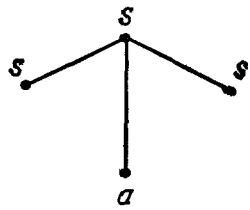
и имеет другую структуру: $((b)a((b)a(b)))$.

Данный результат можно обобщить.

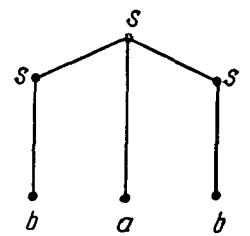
В самом деле, множество $\mathfrak{L}_1$, полученное путем аннулирования членов, не являющихся константами, соответствует выводу

(1)

При получении $\mathfrak{L}_2$ появляется «одночлен» $\mathfrak{L}a\mathfrak{L}$; это позволяет присоединить к нетерминальному выводу



терминальные выводы вида (1), в результате чего мы получим



(2)

При получении $\mathfrak{L}_3$ фигурирует тот же нетерминальный вывод, в котором к каждому из висячих S можно присоединить дерево вида (1) или (2).

Заключение. Мы начали изучение языков, задаваемых системами уравнений, встав на сугубо теоретико-множественную точку зрения. Однако тот факт, что мы встретились с проблемами представления структуры цепочек и с неоднозначностью, заставляет нас пересмотреть некоторые понятия и обозначения, а также ввести в рассмотрение формальные степенные ряды (см. § 11.2).

11.1.7. Система уравнений, связанная с КС-грамматикой

Попытаемся обобщить представления и понятия, рассматривавшиеся до сих пор на материале частных случаев.

Пусть имеется КС-грамматика G, порождающая КС-язык L. Предположим для простоты, что ее вспомогательный словарь содержит только три символа: A, B и аксиому S. (Нетрудно будет убедиться, что это предположение не отражается на общности наших рассуждений.) Предположим, далее, что G не содержит ни правил вида $C \to E$, ни правил вида $C \to D$ ($C, D \in V_A$, E — пустая цепочка) и что любой ее нетерминальный символ может рано или поздно перейти в терминальный, т. е. что в G нет «непродуктивных» символов. Это предположение также не сказывается на общности рассуждений, поскольку мы уже знаем, что правила указанного

вида и непродуктивные символы можно устранить без изменения порождающей силы грамматики и даже без изменения (или с весьма незначительным изменением) структур, приписываемых грамматикой порождаемым цепочкам.

Имеется три группы правил грамматики G (в эти группы объединяются правила с одинаковыми левыми частями):

$$1)\ S \to \Phi_1, \quad S \to \Phi_2, \ \ldots, \quad S \to \Phi_k;$$
$$2)\ A \to \Psi_1, \quad A \to \Psi_2, \ \ldots, \quad A \to \Psi_l;$$
$$3)\ B \to \Theta_1, \quad B \to \Theta_2, \ \ldots, \quad B \to \Theta_m;$$

здесь Φ, Ψ и Θ — цепочки в словаре $V = V_A \cup V_T$.

Сопоставим нетерминальным символам S, A и B «языковые» переменные $\mathfrak{L}$, $\mathfrak{A}$ и $\mathfrak{B}$ соответственно. Заменим в цепочках Φ, Ψ и Θ каждое вхождение нетерминального символа вхождением соответствующей переменной, затем произведем объединение всех цепочек Φ в формальное выражение $f = \Phi_1' \cup \ldots \cup \Phi_k'$ [1]), всех цепочек Ψ — в выражение g и цепочек Θ — в выражение h.

Рассмотрим теперь систему уравнений

$$(\sigma)\ :\ \begin{cases} \mathfrak{L} = f(\mathfrak{L}, \mathfrak{A}, \mathfrak{B}), \\ \mathfrak{A} = g(\mathfrak{L}, \mathfrak{A}, \mathfrak{B}), \\ \mathfrak{B} = h(\mathfrak{L}, \mathfrak{A}, \mathfrak{B}). \end{cases}$$

Примеры. 1) Для языка, порождаемого грамматикой

$$V_A = \{S, A, B\}, \qquad V_T = \{a, b, c\},$$
$$\{S \to aSa,\ S \to bSb,\ S \to c\},$$

мы получим уравнение, рассмотренное выше.

2) Для грамматики

$$\left|\begin{array}{l} S \to AB \\ A \to Sc \\ B \to dB \\ A \to a \\ B \to b \end{array}\right|$$

мы получим следующую систему уравнений:

$$\begin{cases} \mathfrak{L} = \mathfrak{A}\mathfrak{B}, \\ \mathfrak{A} = \mathfrak{L}c \cup a, \\ \mathfrak{B} = d\mathfrak{B} \cup b. \end{cases}$$

[1]) Здесь Φ_i' — выражение, полученное из Φ_i заменой S, A, B на $\mathfrak{L}$, $\mathfrak{A}$, $\mathfrak{B}$ соответственно. — *Прим. ред.*

Заметим, что аксиома участвует в этих уравнениях «на тех же правах», что и остальные вспомогательные символы. Переменные $\mathfrak{A}$ и $\mathfrak{B}$ играют роль, аналогичную роли $\mathfrak{L}$: они соответствуют языкам, порождаемым грамматиками, имеющими те же словари и правила, что и данная, но другие аксиомы: в одной грамматике аксиомой является A, в другой — B.

Неизвестным в нашей системе уравнений является тройка языков (L_S, L_A, L_B), грамматики которых имеют в качестве аксиом символы S, A и B соответственно.

Решая эту систему, мы будем действовать точно так же, как в разобранном выше частном случае. Во всяком решении каждый из трех языков обязательно содержит цепочки, играющие роль «свободных членов» правой части соответствующего уравнения.

Исходя из начальной тройки

$$(\mathfrak{L}_0 = \omega,\ \mathfrak{A}_0 = \omega,\ \mathfrak{B}_0 = \omega),$$

мы будем последовательно образовывать тройки, входящие в решение:

$$[\mathfrak{L}_1 = f(\mathfrak{L}_0, \mathfrak{A}_0, \mathfrak{B}_0);\quad \mathfrak{A}_1 = g(\mathfrak{L}_0, \mathfrak{A}_0, \mathfrak{B}_0);\quad \mathfrak{B}_1 = h(\mathfrak{L}_0, \mathfrak{A}_0, \mathfrak{B}_0)];$$

. . . . . .

$$[\mathfrak{L}_n = f(\mathfrak{L}_{n-1}, \mathfrak{A}_{n-1}, \mathfrak{B}_{n-1});\quad \mathfrak{A}_n = g(\mathfrak{L}_{n-1}, \ldots);\quad \mathfrak{B}_1 = h(\mathfrak{L}_{n-1}, \ldots)];$$

. . . . . .

В качестве «индуктивного предела» мы получим $[\mathfrak{L}, \mathfrak{A}, \mathfrak{B}]$, где $\mathfrak{L} = \mathfrak{L}_1 \cup \mathfrak{L}_2 \cup \ldots$, $\mathfrak{A} = \mathfrak{A}_1 \cup \mathfrak{A}_2 \cup \ldots$, $\mathfrak{B} = \mathfrak{B}_1 \cup \mathfrak{B}_2 \cup \ldots$, т. е. единственную тройку — наименьшее решение системы. Нетрудно показать, что его компоненты совпадут с языками, порождаемыми по правилам грамматики G из аксиом S, A и B соответственно[1]).

11.1.8. Пример решения системы уравнений

Решим в качестве примера систему уравнений, выписанную выше:

$$\begin{cases} \mathfrak{L} = \mathfrak{A}\mathfrak{B}, \\ \mathfrak{A} = \mathfrak{L}c \cup a, \\ \mathfrak{B} = d\mathfrak{B} \cup b. \end{cases}$$

В промежуточных результатах наших выкладок мы будем сохранять расстановку скобок, индуцированную ходом решения. При

[1]) Можно показать также, что (при условии отсутствия в исходной грамматике правил вида $A \to E$ и $A \to B$) наименьшее решение является единственным. Доказательство аналогично приведенному в примечании [1]) на стр. 117 для частного случая. — *Прим. ред.*

этом мы будем помечать левую фигурную скобку, возникающую в i-м уравнении ($i = 1, 2, 3$), цифрой i; этой же цифрой будут затем помечаться (левые) круглые скобки, «происходящие» от данной фигурной.

Итак, мы имеем:

$$\begin{cases} \mathfrak{L}_0 = \omega, \\ \mathfrak{A}_0 = \omega, \\ \mathfrak{B}_0 = \omega; \end{cases}$$

$$\begin{cases} \mathfrak{L}_1 = \underset{1}{\{}\omega\omega\} = \omega, \\ \mathfrak{A}_1 = \underset{2}{\{}\omega c \cup a\} = \underset{2}{\{}a\}, \\ \mathfrak{B}_1 = \underset{3}{\{}d\omega \cup b\} = \underset{3}{\{}b\}; \end{cases}$$

$$\begin{cases} \mathfrak{L}_2 = \underset{1}{\{}\underset{2}{\{}a\}\underset{3}{\{}b\}\} = \underset{1}{\{}\underset{2}{(}a\underset{3}{)}\underset{3}{(}b\underset{3}{)}\}, \\ \mathfrak{A}_2 = \underset{2}{\{}\omega c \cup a\} = \underset{2}{\{}a\}, \\ \mathfrak{B}_2 = \underset{3}{\{}d\,\underset{3}{\{}b\}\cup b\} = \underset{3}{\{}\underset{3}{(}d\underset{3}{(}b\underset{3}{))},\ b\}; \end{cases}$$

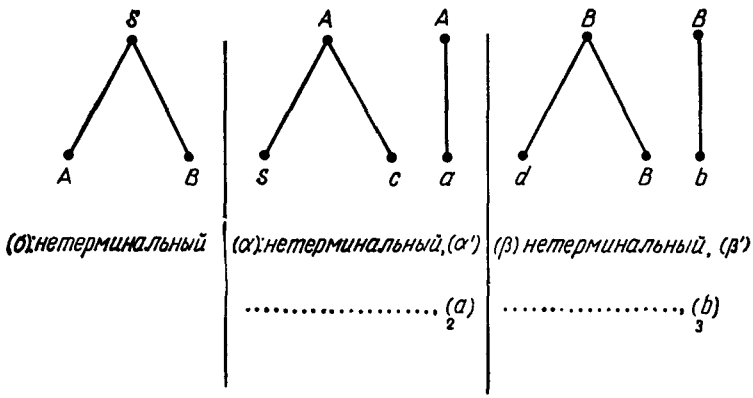
$$\begin{cases} \mathfrak{L}_3 = \underset{1}{\{}\underset{2}{\{}a\}\underset{3}{\{}\underset{3}{(}d\underset{3}{(}b\underset{3}{))},\ b\}\} = \underset{1}{\{}\underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}d\underset{3}{(}b\underset{3}{)))},\ \underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}b\underset{3}{))}\}, \\ \mathfrak{A}_3 = \underset{2}{\{}\underset{1}{\{}\underset{2}{(}a\underset{3}{)}\underset{3}{(}b\underset{3}{)}\}c \cup a\} = \underset{2}{\{}\underset{2}{(}\underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}b\underset{3}{))}c\underset{2}{)},\ a\}, \\ \mathfrak{B}_3 = \underset{3}{\{}d\,\underset{3}{\{}\underset{3}{(}d\underset{3}{(}b\underset{3}{))},\ b\}\cup b\} = \underset{3}{\{}\underset{3}{(}d\underset{3}{(}d\underset{3}{(}b\underset{3}{))},\ \underset{3}{(}d\underset{3}{(}b\underset{3}{))},\ b\}; \end{cases}$$

$$\begin{cases} \mathfrak{L}_4 = \underset{1}{\{}\underset{2}{\{}\underset{2}{(}\underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}b\underset{3}{))}c\underset{2}{)},\ a\}\underset{3}{\{}\underset{3}{(}d\underset{3}{(}d\underset{3}{(}b\underset{3}{)))},\ \underset{3}{(}d\underset{3}{(}b\underset{3}{))},\ b\}\} = \\ \quad = \underset{1}{\{}\underset{1}{(}\underset{1}{(}\underset{2}{(}\underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}b\underset{3}{))}c\underset{2}{)}\underset{3}{(}d\underset{3}{(}d\underset{3}{(}b\underset{3}{))))},\ \underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}d\underset{3}{(}d\underset{3}{(}b\underset{3}{))))}, \\ \quad \underset{1}{(}\underset{2}{(}\underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}b\underset{3}{))}c\underset{2}{)}\underset{3}{(}d\underset{3}{(}b\underset{3}{))}\underset{1}{)},\ \underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}d\underset{3}{(}b\underset{3}{))}\underset{1}{)},\ \underset{1}{(}\underset{2}{(}\underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}b\underset{3}{))}c\underset{2}{)}b\underset{1}{)},\ \underset{1}{(}\underset{2}{(}a\underset{3}{)}\underset{3}{(}b\underset{3}{))}\underset{1}{)}\}, \\ \mathfrak{A}_4 = \ldots \\ \mathfrak{B}_4 = \ldots. \end{cases}$$

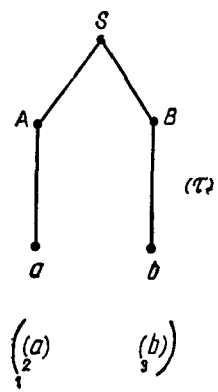
В результате мы одновременно получим скобочные представления структур цепочек языков, порожденных по правилам грамма-

тики G из аксиом S, A и B. Сопоставим эти скобочные представления структур со структурами, которые приписывает данным цепочкам сама грамматика G.

На первом уровне мы имеем следующие выводы:

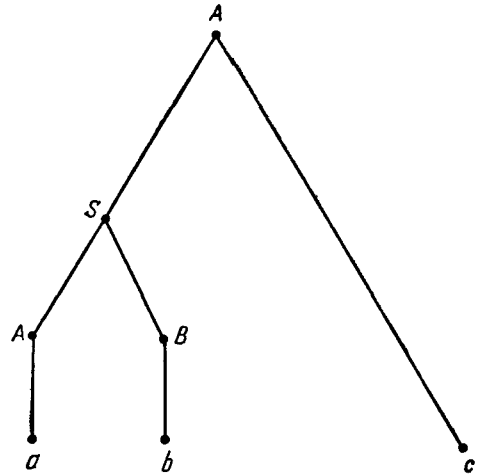


На втором уровне деревья (α') и (β') подвешиваются к узлам A и B дерева нетерминального вывода (σ):



На третьем уровне мы имеем для языка, порождаемого из аксиомы A, дерево, представляющее собой результат присоединения

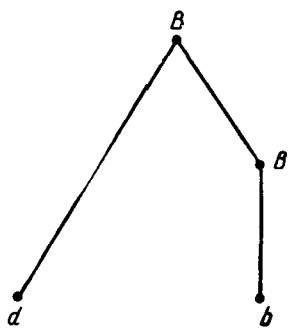
дерева (τ) к узлу S дерева (α):



Это дерево отвечает скобочной структуре

$$\underset{2}{\Big(}\underset{1}{\big(}\underset{2}{(a)}\quad\underset{3}{(b)}\big)\quad c\Big)$$

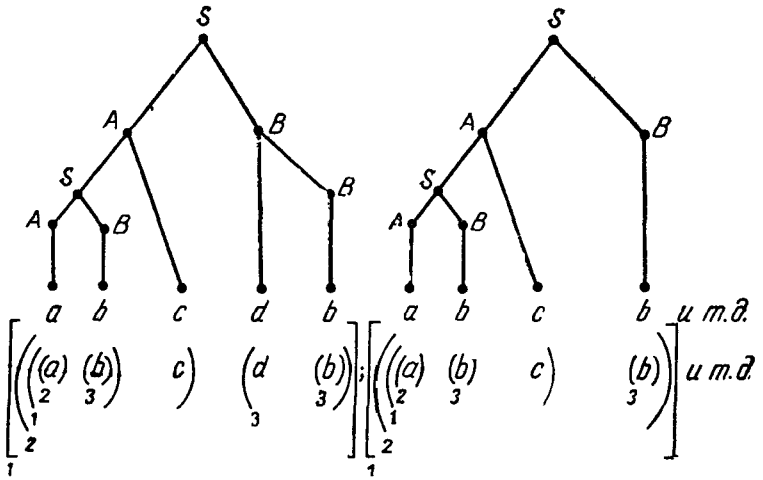
Аналогичным образом получается (для языка, порождаемого из аксиомы B, на втором уровне) следующее дерево — результат присоединения терминального дерева (β') к узлу B нетерминального дерева (β):



то есть

$$\underset{3}{(d}\qquad\underset{3}{(b)}\big)$$

На четвертом уровне для языка, порождаемого из аксиомы S, строятся все деревья, которые можно получить, присоединяя к узлам A и B дерева (σ) все терминальные деревья предыдущих уровней с вершинами A и B:



Мы видим, таким образом, что обе расстановки скобок совпадают. Это объясняется тем, что в ходе решения системы уравнений при каждой итерации снова появлялись простейшие схемы вывода, причем к нетерминальным узлам можно было присоединять терминальные деревья, построенные на предыдущих шагах. Структура цепочек остается той же самой, хотя в процессе решения системы уравнений деревья строятся «снизу вверх», а не «сверху вниз», как при непосредственном использовании правил грамматики.

11.1.9. Заключение

Итак, всякой КС-грамматике G можно поставить в соответствие систему уравнений, число которых равно числу ее нетерминальных символов.

Каждое из этих уравнений имеет вид

$$\mathfrak{L} = f(\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \ldots),$$

где f — многочлен.

Решением данной системы является набор языков, каждый из которых порождается в соответствии с правилами грамматики G из аксиомы, совпадающей с одним из ее нетерминальных символов.

Говорят, что этот набор языков *удовлетворяет* данной грамматике. Класс КС-языков совпадает с классом языков, задаваемых

системами уравнений рассматриваемого вида. В то же время мы видели, что определенная организация процесса решения системы уравнений позволяет обнаружить в цепочках языка некоторую структуру. Это побуждает нас отказаться от чисто теоретико-множественной точки зрения, принятой нами вначале. Что это означает более конкретно, станет ясно в следующем параграфе.

§ 11.2. ЯЗЫКИ И ФОРМАЛЬНЫЕ СТЕПЕННЫЕ РЯДЫ

Стремление описывать последовательности языков $L_0, L_1, L_2, \ldots, L_n, \ldots$ более «алгебраическим» способом и желание учитывать степень неоднозначности цепочек наводят на мысль рассмотреть многочлены, членами которых являлись бы цепочки упомянутых языков, а коэффициентами — целые положительные числа, причем коэффициент при цепочке должен быть равен числу приписываемых этой цепочке структур. Если язык-решение бесконечен, то такие многочлены на достаточно далеких шагах процесса будут иметь члены со сколь угодно высокими степенями.

Однако члены данной фиксированной степени, начиная с некоторого шага, перестают изменяться, что позволяет перейти к пределу и получить тем самым формальные степенные ряды.

Приступая к изучению языков с этой новой точки зрения, мы сформулируем как бы a priori ряд определений, опирающихся в действительности на приведенные выше эмпирические соображения. При этом для большей общности мы примем соглашение, что коэффициенты могут быть целыми числами любого знака; целесообразность этого обобщения станет очевидной ниже.

11.2.1. Степенные ряды

Пусть $V_T = \{a_i \mid 1 \leqslant i \leqslant n\}$ — терминальный словарь. Множество всех цепочек, которые можно составить из символов этого словаря, является счетным.

Сопоставим каждой цепочке $x \in V_T^*$ некоторое целое число. Другими словами, зададим функцию, или отображение, (r) с областью определения V_T^* и множеством значений, содержащимся в множестве всех целых чисел $\mathbf{Z}$:

$$x \xrightarrow{(r)} \langle r, x\rangle \in \mathbf{Z}.$$

С помощью этого отображения мы можем определить *ряд*

$$r = \sum_x \langle r, x\rangle x,$$

в котором коэффициент при каждом члене x равен тому числу, которое отображение (r) сопоставляет данному члену.

Пример. Пусть $V_T = \{a, b\}$. Выписывая члены ряда в порядке возрастания длин, а при равной длине — в лексикографическом порядке, мы получим

$$E,\ a,\ b,\ aa,\ ab,\ ba,\ bb,\ aaa,\ aab,\ aba,\ \ldots.$$

Коэффициент каждого члена определяется так: к числу, равному длине данного члена, прибавим 1 и возьмем остаток от деления этой суммы на 3. Это и есть коэффициент.

В нашем случае получится следующий ряд:

$$r = 1E + 2a + 2b + 0aa + 0ab + 0ba + 0bb + 1aaa + 1aab + 1aba + \ldots.$$

Члены с нулевыми коэффициентами можно не выписывать.

Если (r) отображает V_T^* на (0, 1), т. е. если функция (r) является характеристической, то ряд называется *характеристическим*. Всякому ряду можно сопоставить характеристический ряд, сохранив все нулевые коэффициенты, а ненулевые заменив единицами.

Пусть имеется степенной ряд r; тогда множество цепочек с ненулевыми коэффициентами образует язык L, называемый *опорным множеством* данного ряда и обозначаемый через

$$L = \mathrm{supp}\,(r).$$

11.2.2. Операции над степенными рядами

Множество степенных рядов образует кольцо относительно операций сложения и умножения, которые мы сейчас определим.

Сложение рядов определяется обычным образом: в ряде $r + r'$, представляющем собой сумму рядов r и r', коэффициент при каждой цепочке есть сумма коэффициентов при соответствующих цепочках в рядах r и r':

$$\langle r + r',\ x\rangle = \langle r,\ x\rangle + \langle r',\ x\rangle.$$

Умножение рядов определяется, как обычно. Следует только помнить о его некоммутативности: конкатенация сомножителей выполняется в определенном порядке, без перестановок. В произведении рядов r и r', которое обозначается через rr', каждый коэффициент $\langle rr', x\rangle$ есть сумма всевозможных произведений $\langle r_i,\ y\rangle \langle r'_j,\ z\rangle$, таких, что $yz = x$.

Утверждение о том, что множество степенных рядов имеет структуру кольца, предоставляется доказать самому читателю.

Умножение на целое число. Умножение ряда на целое число также определяется обычным способом. В ряде $n \cdot r$ коэффициент при x равен $n\langle r, x\rangle$.

11.2.3. Топологические понятия

Уточним теперь способ перехода от многочленов к рядам и введем понятие окрестности.

Будем говорить, что ряд r *эквивалентен ряду* r' *по модулю* n, и писать

$$r \equiv r' \pmod{n},$$

если $(l(x) \leqslant n) \Rightarrow \langle r, x\rangle = \langle r', x\rangle$.

Пусть имеется бесконечная последовательность рядов $r_1, r_2, \ldots$, такая, что $r_n \equiv r_{n'}$ для всякого n и для всякого $n' > n$. В этом случае понятно, как определить предел последовательности $r_1, r_2, \ldots$: это предел последовательности многочленов, получаемых путем отбрасывания в каждом r_n всех членов, длины которых превосходят n [1]).

Теперь ясно, как снабдить кольцо рядов топологией: окрестность некоторого ряда будет представлять собой множество всех рядов, эквивалентных ему по некоторому модулю.

11.2.4. Ряды с неотрицательными коэффициентами

Ряды с неотрицательными коэффициентами (*«положительные ряды»*) образуют полукольцо. Кроме того, их опорные множества обладают следующими важными свойствами.

Сложение. Коэффициент $\langle r + r', x\rangle$, представляющий собой сумму двух неотрицательных целых чисел, отличен от нуля тогда и только тогда, когда хотя бы одно из этих чисел не равно нулю.

Для положительных рядов опорное множество суммы равно объединению опорных множеств слагаемых.

Умножение. Если опорное множество ряда r содержит цепочку y, а опорное множество ряда r' — цепочку z, то опорное множество ряда rr' содержит цепочку yz (поскольку аннулирование членов невозможно). Обратно, если опорное множество ряда rr' содержит цепочку x, то существует хотя бы одна пара (y, z), где y и z принадлежат опорным множествам рядов r и r' соответственно, такая, что $yz = x$.

Следовательно,

для положительных рядов опорное множество произведения равно произведению (в смысле умножения языков) опорных множеств.

11.2.5. Положительные ряды и КС-грамматики

Читатель, вероятно, уже заметил, что ряды с целыми неотрицательными коэффициентами позволяют дать простое описание выкладок, к которым мы прибегали, строя и решая систему уравнений, связанную с данной КС-грамматикой.

Будем теперь считать, что переменные $\mathfrak{L}$, $\mathfrak{A}$, $\mathfrak{B}$, ... представляют не языки в теоретико-множественном смысле, а формальные

[1]) Заметим, что в этой последовательности каждый следующий член есть продолжение предыдущего.

ряды. Каждому ряду отвечает свое опорное множество, являющееся языком, а также коэффициенты, характеризующие принадлежность цепочки языку и степень ее неоднозначности:

0 — данная цепочка не порождается,

1 — данная цепочка порождается одним способом,

2 — данная цепочка порождается двумя разными способами

и т. д.

Система уравнений, сопоставленная данной КС-грамматике, записывается по-прежнему в виде

$$\mathfrak{L} = f(\mathfrak{L}, \mathfrak{A}, \mathfrak{B}, \ldots),$$
$$\mathfrak{A} = g(\mathfrak{L}, \mathfrak{A}, \mathfrak{B}, \ldots),$$
$$\mathfrak{B} = h(\mathfrak{L}, \mathfrak{A}, \mathfrak{B}, \ldots),$$

где f, g, h, ... суть многочлены, все коэффициенты которых равны 1.

Предложенный выше метод годится для решения системы также и при новой интерпретации. Последовательные приближения

$$(\mathfrak{L}_0, \mathfrak{A}_0, \mathfrak{B}_0, \ldots), (\mathfrak{L}_1, \mathfrak{A}_1, \mathfrak{B}_1, \ldots), \ldots$$

— это формальные ряды специального вида, а именно многочлены. По модулю n, где n — максимальная длина их членов, эти многочлены эквивалентны искомому решению, к которому они стремятся как к индуктивному пределу.

В качестве упражнения читатель может переписать в новых обозначениях примеры, которые мы рассматривали в предыдущем разделе.

Здесь мы приведем новый пример, который интересен и сам по себе.

Пример. Выше мы рассматривали неоднозначную грамматику с правилами

$$|\, S \to SaS,\ S \to b \,|.$$

Напротив, грамматика с правилами

$$|\, S \to aSS,\ S \to b \,|,$$

соответствующая бесскобочной записи (тех же — в содержательном смысле — формул!), является однозначной. Следовательно, соответствующий ряд — характеристический (т. е. все его коэффициенты равны 0 и 1). Чтобы получить этот ряд, выпишем уравнение

$$\mathfrak{L} = a\mathfrak{L}\mathfrak{L} + b,$$

решая которое, получаем

$\mathfrak{L}_0 = \omega$,

$\mathfrak{L}_1 = b$,

$\mathfrak{L}_2 = a(b)(b) + b$;

$$\mathfrak{L}_3 = a[a(b)(b)+b][a(b)(b)+b]+b =$$
$$= a[a(b)(b)][a(b)(b)]+a[b][a(b)(b)]+a[a(b)(b)][b]+a[b][b]+b;$$

теперь опустим скобки:

$$L_2 = ab^2+b;$$
$$L_3 = a^2b^2ab^2+abab^2+a^2b^3+ab^2+b.$$

Мы видим, что ряд действительно является характеристическим.

Заключение. КС-грамматикам могут быть сопоставлены системы уравнений, вторые члены которых суть многочлены с коэффициентами, равными 1 [1]). КС-языки являются опорными множествами положительных рядов, полученных в результате решения подобных систем; коэффициенты этих рядов показывают степень неоднозначности цепочек (т. е. число сопоставляемых им структур).

Остается выяснить, можно ли дать разумную лингвистическую интерпретацию рядам с коэффициентами любого знака и системам уравнений с не обязательно положительными коэффициентами. Мы рассмотрим этот вопрос в гл. XVI, посвященной алгебраическим языкам.

11.2.6. Упражнения

1. Пусть имеется язык L_m, порождаемый следующей нормальной грамматикой:

$$V_A = \{S, A, B\};\quad V_T = \{a, b, c\};\quad \text{аксиома} - S;$$
$$\left|\begin{array}{l} S \to aA \\ S \to bB \\ S \to c \\ A \to Sa \\ B \to Sb \end{array}\right|.$$

Построить соответствующую этой грамматике систему уравнений и решить ее с помощью формальных рядов. Исследовать индуцированную структуру цепочек, сравнив ее со структурой, которую дает грамматика, рассмотренная в п. 11.1.5.

2. Пусть имеется грамматика

$$\left|\begin{array}{l} S \to SS \\ S \to a \\ S \to b \end{array}\right|$$

с аксиомой S и терминальным алфавитом $\{a, b\}$.

Решить уравнение, соответствующее этой грамматике. Исследовать индуцированную структуру и неоднозначность цепочек. Обнаружить закон, по которому определяется степень неоднозначности.

[1]) Это следует просто из того, что в грамматике нет двух одинаковых правил. — *Прим. ред.*

Глава XII

ГРАММАТИКИ НЕПОСРЕДСТВЕННО СОСТАВЛЯЮЩИХ. АВТОМАТЫ С ЛИНЕЙНО ОГРАНИЧЕННОЙ ПАМЯТЬЮ

§ 12.1. ГРАММАТИКИ НЕПОСРЕДСТВЕННО СОСТАВЛЯЮЩИХ (НС-ГРАММАТИКИ) [1]

12.1.1. Содержательное (лингвистическое) обоснование

Рассмотрим простую фразу, построенную по схеме $Gn_1\ V\ Gn_2$ (группа подлежащего + сказуемое + группа дополнения: *Мальчик ест яблоки*, *Мешок весит пуд*, ...). Фиксируем семантический класс существительных, выступающих в функции вершин групп подлежащего и дополнения: названия одушевленных существ, названия абстрактных понятий и т. п.; тогда семантический класс глагола, который может быть сказуемым, зависит от этих семантических классов, т. е. класс допустимых V *ограничен* контекстом Gn_1—Gn_2.

Подобными ситуациями изобилуют и естественные, и искусственные языки; поэтому формализация указанного явления представляет очевидный интерес.

12.1.2. НС-грамматики

НС-грамматика — это совокупность следующих трех **объектов**:

. конечный алфавит $V = V_A \cup V_T$; $V_A \cap V_T = \emptyset$;

.. аксиома $S \in V_A$;

... множество правил вида

$$\varphi_1 A \varphi_2 \to \varphi_1 \omega \varphi_2,$$

где $A \in V_A$, φ_1, $\varphi_2 \in V^*$ и ω не пусто.

Подобные правила, интерпретируемые следующим образом: «в контексте φ_1 (*левый контекст*) и контексте φ_2 (*правый контекст*) нетерминальный символ A заменяется цепочкой ω», мы будем называть *НС-правилами* [2]).

Заметим, что КС-грамматики образуют подкласс НС-грамматик, соответствующий случаю, когда φ_1 и φ_2 пусты.

---

[1]) В оригинале — grammaires à règles contextuelles, т. е. «грамматики с контекстными правилами». Этот термин представляется нам неудачным, поскольку если мы примем его, то контекстно-свободные, или «бесконтекстные», грамматики окажутся частным случаем «контекстных». Принятый в русской литературе термин «грамматики непосредственно составляющих» (ср. англ phrase-structure grammars) объясняется тем, что грамматики этого класса сопоставляют порождаемым ими цепочкам системы (непосредственно) составляющих точно таким же образом, как являющиеся их частным случаем КС-грамматики. — *Прим. ред.*

[2]) В оригинале — английский термин context-sensitive, т. е. «контекстно-чувствительные»; ср. предыдущее примечание. — *Прим. ред.*

12.1.3. Одно важное свойство НС-грамматик

В силу требования, чтобы ω никогда не было пусто, всякая НС-грамматика является полутуэвской системой специального вида, продукции которой

$$P\varphi_1 A\varphi_2 Q \to P\varphi_1\omega\varphi_2 Q$$

удовлетворяют условию *неубывания длины*. Рассмотрим это условие отдельно.

12.1.4. Условие неубывания длины

Пусть имеется полутуэвская система в алфавите $V = V_A \cup V_T$; пусть ее продукции

$$Pg_iQ \to P\bar{g}_iQ$$

удовлетворяют *условию неубывания длины*:

$$|\bar{g}_i| \geqslant |g_i|.$$

Каждую такую продукцию (правило) можно заменить последовательностью НС-правил вида $\varphi_1 A\varphi_2 \to \varphi_1\omega\varphi_2$. Делается это так. Пусть

$$\left.\begin{aligned} g_i &= \alpha_1 \dots \alpha_k \\ \bar{g}_i &= \beta_1 \dots \beta_k \dots \beta_{k+q} \end{aligned}\right| \quad \alpha_i, \beta_i \in V;\ q \geqslant 0.$$

Введем новые (вспомогательные) символы $\gamma_1, \dots, \gamma_k$ и рассмотрим правила

$$\begin{aligned}
&\alpha_1\alpha_2\alpha_3 \dots \alpha_{k-1}\alpha_k \to \gamma_1\alpha_2\alpha_3 \dots \alpha_{k-1}\alpha_k,\\
&\gamma_1\alpha_2\alpha_3 \dots \alpha_{k-1}\alpha_k \to \gamma_1\gamma_2\alpha_3 \dots \alpha_{k-1}\alpha_k,\\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
&\gamma_1\gamma_2\gamma_3 \dots \gamma_{k-1}\alpha_k \to \gamma_1\gamma_2\gamma_3 \dots \gamma_{k-1}\gamma_k,\\
&\gamma_1\gamma_2\gamma_3 \dots \gamma_{k-1}\gamma_k \to \beta_1\gamma_2\gamma_3 \dots \gamma_{k-1}\gamma_k,\\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
&\beta_1 \dots \beta_{k-2}\gamma_{k-1}\gamma_k \to \beta_1 \dots \beta_{k-2}\beta_{k-1}\gamma_k,\\
&\beta_1 \dots \beta_{k-2}\beta_{k-1}\gamma_k \to \beta_1 \dots \beta_{k-2}\beta_{k-1}\beta_k \dots \beta_{k+q}.
\end{aligned}$$

Все эти правила имеют вид

$$\varphi_1 A\varphi_2 \to \varphi_1\omega\varphi_2, \quad |\omega| > 0.$$

В то же время ясно, что последовательное применение этих правил равносильно применению правила $g_i \to \bar{g}_i$. Можно также показать, что вместо применения этих правил в произвольном порядке их можно применять подряд и в указанном здесь порядке.

Таким образом, для всякой полутуэвской системы, продукции которой удовлетворяют условию неубывания длины, можно

построить эквивалентную НС-грамматику. В результате можно сформулировать следующую теорему:

Теорема. *Класс языков, порождаемых НС-грамматиками, совпадает с классом языков, порождаемых грамматиками, удовлетворяющими условию неубывания длины.*

НС-грамматика, все правила которой удовлетворяют условию неубывания длины, называется *неукорачивающей.*

12.1.5. Рекурсивность языков, порождаемых неукорачивающими грамматиками

Предложение. *Всякая неукорачивающая грамматика G порождает рекурсивный язык.*

Доказательство. При фиксированном алфавите V данное предложение очевидно, если во всех правилах длина правой части строго больше длины левой части. Допустим, что в некоторых правилах длина правой части равна длине левой части. Множество всех цепочек длины n, порождаемых грамматикой G, имеет конечную мощность $N = N(n)$, причем $N = N(n) \leqslant [\mathrm{card}(V)]^n$ [1]).

Возьмем какую-нибудь цепочку из n букв и проведем все те выводы в G, начинающиеся с этой цепочки, которые не увеличивают длину. В каждом из таких выводов после $(N+1)$-го шага, или даже раньше, повторно встретится какая-либо цепочка длины n, уже встречавшаяся ранее: в выводе образуется *петля.* Подобные петли можно выкинуть, сохраняя лишь «полезную» часть вывода. Отсюда ясно, что для любого m в грамматике G имеется лишь конечное число таких «полезных» выводов цепочек f из аксиомы, что $|f| \leqslant m$; именно, число этих выводов не превосходит $N(1) + \ldots + N(m)$. Эти выводы можно перебрать и проверить, принадлежит ли f языку $L(G)$; тем самым наше предложение доказано.

§ 12.2. ЛИНЕЙНО ОГРАНИЧЕННЫЕ АВТОМАТЫ

12.2.1. Определение

Линейно ограниченный автомат (сокращенно: *ЛО-автомат*) — это система следующих объектов:

. *Управляющее устройство*, принимающее конечное число *состояний* S_j, среди которых выделено *начальное состояние* S_0 и подмножество Φ *заключительных состояний.*

.. *Лента*, разбитая на ячейки, в которых пишутся (и с которых считываются) символы *алфавита* V, дополненного специальным *граничным маркером* #; этот маркер ставится в начале и в конце всякой рабочей цепочки.

---

[1]) card (V) означает мощность V.

... *Головка*, выполняющая операции трех типов: 1) запись символа на ленте, 2) считывание символа с ленты, 3) перемещение вдоль ленты влево или вправо. Эти операции выполняются в соответствии с *командами* вида

$$(a_i,\ S_j) \rightarrow (S_k,\ a_l,\ M),$$

означающими: «Если управляющее устройство находится в состоянии S_j и головка читает на ленте символ a_i, то управляющее устройство переходит в состояние S_k, головка пишет на ленте вместо символа a_i символ a_l, и

— если $M = 0$, остается на месте;

— если $M = +1$, смещается на одну ячейку вправо;

— если $M = -1$, смещается на одну ячейку влево».

Если $a_i = \#$, то и $a_l = \#$ [1]).

12.2.2. Вычисления

Если на ленте ЛО-автомата написать цепочку символов из V, окаймленную маркерами $\#$, установить головку так, чтобы она обозревала левый маркер, и привести управляющее устройство в начальное состояние S_0, то автомат начинает выполнять команды и делает это до тех пор, пока головка не дойдет до правого маркера. Если в этот момент управляющее устройство оказывается в одном из заключительных состояний, то исходная цепочка является по определению допустимой; в противном случае она не является допустимой.

Рассматриваются как детерминированные, так и недетерминированные ЛО-автоматы.

12.2.3. Языки, допускаемые ЛО-автоматами

Множество всех допустимых цепочек есть по определению язык, *допускаемый* ЛО-автоматом.

Относительно языков, допускаемых ЛО-автоматами, имеет место следующее

Предложение. *Для всякого ЛО-автомата* $\mathfrak{A}$ *можно построить неукорачивающую грамматику* G, *такую, что* $L(G) = L(\mathfrak{A})$.

Доказательство. В качестве терминального словаря V_T грамматики G мы возьмем алфавит автомата $\mathfrak{A}$, а в качестве ее вспомогательного словаря — множество

$$V_A^G = \{A_{ij},\ S,\ T,\ \#\},$$

где

[1]) Таким образом, ЛО-автомат — это не что иное, как недетерминированная машина Тьюринга (перемещение головки, очевидно, равносильно использовавшемуся в гл IV перемещению ленты), на которую наложено дополнительное ограничение: в процессе работы не увеличивать длину рабочей цепочки Именно для этого нужны окаймляющие цепочку маркеры, сохраняющиеся в течение всего вычисления. — *Прим ред.*

. A_{ij} сопоставляется ситуации (a_i, S_j);
.. S — аксиома грамматики G (новый символ);
... T — еще один новый вспомогательный символ;
.... $\#$ — маркер, с помощью которого отмечаются начало и конец цепочки.

Правила грамматики G определенным образом связаны с ситуациями автомата $\mathfrak{A}$: они выполняют его вычисления, но только в обратном направлении — справа налево. Эти правила распадаются на три группы:

I. Правила группы I порождают цепочки вида

$$\varphi_T A_{if},$$

где $\varphi_T \in V_T^*$ и A_{if} соответствует заключительной ситуации (a_i, S_f). Эти правила таковы:

$S \to TA_{if}$ для всякой заключительной ситуации (a_i, S_f);

$$\left.\begin{array}{l} T \to Ta_l \\ T \to a_l \end{array}\right\} \text{ для всякого } a_l.$$

II. Правила этой группы перерабатывают цепочки, порожденные правилами группы I; их выполнение приводит к терминальной цепочке тогда и только тогда, когда $\mathfrak{A}$ допускает эту цепочку. Проверка допустимости цепочки заканчивается, когда получена ситуация (a_i, S_0), т. е. с выработкой символа A_{i0}.

. Команде $(a_i, S_j) \to (S_k, a_l, +1)$ соответствуют правила $a_l A_{rk} \to A_{ij} a_r$ для всякого $a_r \in V_T$.

.. Команде $(a_i, S_j) \to (S_k, a_l, -1)$ соответствуют правила $A_{rk} a_l \to a_r A_{ij}$ для всякого $a_r \in V_T$.

... Команде $(a_i, S_j) \to (S_k, a_l, 0)$ соответствует правило $A_{lk} \to A_{ij}$.

III. Правила группы III позволяют завершать выводы. Они имеют вид

$$\# A_{i0} \to \# a_i.$$

Ясно, что $L(G) = L(\mathfrak{A})$. При этом G — неукорачивающая грамматика специального типа: левые и правые части ее правил имеют длину 1 или 2.

12.2.4. НС-грамматики и ЛО-автоматы

Можно доказать и обратное утверждение: для всякой неукорачивающей грамматики можно построить ЛО-автомат — вообще говоря, недетерминированный, — допускающий тот же язык.

Идея построения этого автомата состоит в том, что он должен искать в перерабатываемой цепочке вхождения правых частей правил грамматики (помечая принадлежащие этим вхождениям символы какими-либо метками), а затем, найдя такое вхождение, заменять его, символ за символом, вхождением левой части того же правила; при этом длина цепочки не увеличится, так как левая

часть не длиннее правой. Автомат допустит те и только те цепочки, которые он сможет таким способом переработать в аксиому.

Теперь из результата предыдущего пункта с учетом теоремы п. 12.1.4 получается следующая

Теорема (Курода — Ландвебер). *Класс языков, порождаемых НС-грамматиками (класс НС-языков), совпадает с классом языков, допускаемых недетерминированными ЛО-автоматами.*

Неизвестно, совпадает ли этот класс с классом языков, допускаемых детерминированными ЛО-автоматами.

12.2.5. Некоторые свойства НС-языков

Можно доказать, что класс НС-языков замкнут относительно объединения, пересечения, умножения и итерации. Замкнут ли он относительно дополнения, неизвестно.

В то же время класс языков, допускаемых детерминированными ЛО-автоматами, образует булеву алгебру.

12.2.6. Алгоритмические проблемы

Почти все интересные алгоритмические проблемы для НС-грамматик рекурсивно неразрешимы; в частности, неразрешимы проблемы распознавания пустоты, конечности, бесконечности и «контекстной свободности» языка, порождаемого данной НС-грамматикой.

§ 12.3. КЛАССИФИКАЦИЯ АВТОМАТОВ

12.3.1. Линейно ограниченная память

Термин «линейно ограниченный автомат», предложенный Дж. Майхиллом [1]), объясняется тем, что в ходе вычислений такой автомат использует память ограниченного объема.

В предыдущих главах мы рассмотрели три класса автоматов: машины Тьюринга, автоматы с магазинной памятью, конечные автоматы.

Машина Тьюринга располагает на своей ленте неограниченным запасом пустых ячеек, благодаря чему она может запомнить неограниченное число символов. Конечный автомат, напротив, использует память конечного и, более того, фиксированного объема, полностью «умещающуюся» в управляющем устройстве: лента служит здесь исключительно для считывания данных (писать на ней нельзя).

ЛО-автоматы занимают промежуточное положение: объем их памяти не является раз навсегда фиксированным, но и не является неограниченным: он ограничен длиной обрабатываемой цепочки. Что же касается МП-автоматов, то они характеризуются ограниче-

[1]) См. Майхилл [1960]. — *Прим. ред.*

нием, налагаемым не на объем памяти, а на способ ее использования; однако для всякого МП-автомата можно построить эквивалентный ему МП-автомат, у которого ни в одном вычислении длина используемой части рабочей ленты не превышает длины входной цепочки; такой МП-автомат можно рассматривать «почти» как частный случай ЛО-автомата («почти» потому, что МП-автомат имеет отдельную входную ленту, которой лишен ЛО-автомат; впрочем, это различие не является существенным, если мы интересуемся только объемом памяти).

12.3.2. Иерархия автоматов

Таким образом, критерий объема памяти, необходимого для проведения вычисления, позволяет дать следующую иерархическую классификацию автоматов:

Чтобы сделать эту классификацию более тонкой, — в частности, чтобы ввести в нее естественным образом МП-автоматы, — необходимо учитывать не только объем, но и *структуру* используемой при вычислении памяти. На этом пути можно определить ряд интересных подклассов автоматов.

Другое направление, в котором развивается теория автоматов, — это изучение автоматов, работающих в реальном масштабе времени, и вообще изучение классификаций автоматов по времени вычисления [1]).

[1]) Объем памяти и время работы могут рассматриваться как некоторые характеристики сложности вычисления в автомате. Существуют и другие характеристики сложности вычисления. Теория сложности вычислений, развитая

Еще один важный критерий — детерминированность/недетерминированность автомата.

§ 12.4. УПРАЖНЕНИЯ

1. Построить ЛО-автоматы, допускающие языки

$$\{a^n b^n a^n \mid n > 0\},\ \{xcx \mid x \in \{a, b\}^*\};$$

можно ли построить для этих языков детерминированные ЛО-автоматы?

2. Показать, что с помощью НС-правил можно выполнять перестановки; например, если имеется НС-грамматика G, порождающая язык, каждая цепочка которого имеет вид xcy, где x и y не содержат c, то можно построить НС-грамматику G', такую, что $L(G') = \{ycx \mid xcy \in L(G)\}$.

3. Пусть имеются ЛО-автоматы $\mathfrak{A}_1$ и $\mathfrak{A}_2$; построить ЛО-автомат $\mathfrak{A}'$, допускающий язык $L(\mathfrak{A}_1) \cup L(\mathfrak{A}_2)$, и ЛО-автомат $\mathfrak{A}''$, допускающий язык $L(\mathfrak{A}_1) \cap L(\mathfrak{A}_2)$.

Указание. Автомат $\mathfrak{A}''$ можно строить следующим образом. Рядом с любой входной цепочкой f пишем цепочку f', которая получается из f путем копирования ее символов и добавления к ним штрихов; затем та «часть» автомата $\mathfrak{A}$, которая «совпадает с $\mathfrak{A}_1$», проверяет допустимость цепочки f относительно $\mathfrak{A}_1$, а другая часть автомата $\mathfrak{A}$, «идентичная автомату $\mathfrak{A}_2$» с точностью до штрихов, проверяет допустимость цепочки f' относительно $\mathfrak{A}_2$; завершение построения очевидно.

в работах Б. А. Трахтенброта, Г. С. Цейтина, М. Рабина, М. Блюма и др., представляет собой один из важнейших разделов современной теории автоматов. Основные понятия и факты этой теории изложены в книге Б. А. Трахтенброта «Сложность алгоритмов и вычислений», Новосибирск, 1967. — *Прим. ред.*

Часть III
АЛГЕБРАИЧЕСКИЙ ПОДХОД

Глава XIII
ГОМОМОРФИЗМЫ ПОЛУГРУПП

В этой главе излагаются некоторые алгебраические понятия, **которые не раз** понадобятся нам в ходе дальнейшего изложения

§ 13.1. ПРОИЗВОЛЬНЫЕ ПОЛУГРУППЫ

13.1.1. О единичном элементе

Понятие полугруппы было определено в п. 1.1.3, и читатель вероятно, заметил, что всякая свободная полугруппа обязательно обладает единичным элементом. Нам будет удобно предположить, что и всякая полугруппа обладает таким элементом. Если же в данной полугруппе $\mathbf{M}$ единичного элемента нет, то мы всегда можем добавить к ней новый элемент E, такой, что

$$(\forall X \in \mathbf{M})\ [EX = XE = E];$$

$$EE = E.$$

В результате получится полугруппа (в том, что при добавлении E сохраняется ассоциативность, убедиться нетрудно), имеющая единичный элемент E и, очевидно, включающая в себя исходную полугруппу.

13.1.2. Естественный гомоморфизм

Пусть $\mathbf{M} = \{A, B, C, \ldots\}$ — полугруппа с единицей и $\mathfrak{R}$ — отношение эквивалентности на $\mathbf{M}$, совместимое влево и вправо с операцией композиции, т. е. конгруэнция. В гл. I было показано, что $\mathfrak{R}$ определяет разбиение $\mathbf{M}$ на непересекающиеся классы, составляющие фактормножество $\mathbf{M}/\mathfrak{R}$, причем классы $\bar{A}$, $\bar{B}$, $\bar{C}$, ... образуют относительно индуцированной операции новую полугруппу с единицей. Там же рассматривался пример, в котором $\mathbf{M}$ была свободной полугруппой и $\mathfrak{R}$ — эквивалентностью в смысле Туэ.

Рассмотрим *естественное* (или *каноническое*) отображение ψ, связанное с $\mathfrak{R}$; каждому $A \in \mathbf{M}$ отображение ψ ставит в соответствие его класс $\bar{A}$ в разбиении, индуцированном отношением $\mathfrak{R}$.

Для конгруэнции имеет место соотношение

$$[A \xrightarrow{\psi} \bar{A} \,\&\, B \xrightarrow{\psi} \bar{B}] \Rightarrow [AB \xrightarrow{\psi} \bar{A}\bar{B} = \overline{AB}],$$

т. е. образ композиции есть композиция образов.

Отображение ψ называется *естественным гомоморфизмом*, ассоциированным с конгруэнцией $\mathfrak{R}$.

Пример. Пусть $\mathbf{M}$ — свободная полугруппа над алфавитом $\mathfrak{A} = \{a, b, c\}$ и $\mathfrak{R}$ — отношение, соответствующее равенству длин цепочек: пустая цепочка принадлежит классу 0, однобуквенные цепочки — классу 1 и т. п. Фактормножество $\mathfrak{A}^*/\mathfrak{R}$ изоморфно полугруппе с единицей, образованной множеством целых неотрицательных чисел относительно сложения.

13.1.3. Общие определения

Пусть $\mathbf{M}$ и $\mathbf{M}'$ — полугруппы с единицей, φ — отображение первой во (возможно, на) вторую. Говорят, что φ есть *гомоморфизм* $\mathbf{M}$ в (соответственно на) $\mathbf{M}'$, если

$$[A \xrightarrow{\varphi} A' \,\&\, B \xrightarrow{\varphi} B'] \Rightarrow [AB \xrightarrow{\varphi} A'B'].$$

Обозначим через $\mathbf{M}'_1$ множество образов всевозможных элементов $\mathbf{M}$ при отображении φ. Рассмотрим следующее отношение $\mathfrak{R}$ на $\mathbf{M}$: $A_1 \mathfrak{R} A_2$ тогда и только тогда, когда A_1 и A_2 имеют один и тот же образ. Это отношение — эквивалентность; оно дает разбиение множества $\mathbf{M}$ на непересекающиеся классы. Обозначим через $\bar{A}$, $\bar{B}$, $\bar{C}$, ... классы элементов A, B, C, ... соответственно. Очевидно, что существует естественное отображение ψ множества $\mathbf{M}$ на $\mathbf{M}/\mathfrak{R}$, сопоставляющее каждому $A \in \mathbf{M}$ его класс $\bar{A}$.

Кроме того, существует взаимно однозначное соответствие между множеством $\mathbf{M}/\mathfrak{R}$ и множеством $\mathbf{M}'_1$ образов элементов $\mathbf{M}$. Классу $\bar{A}$ всех элементов, образом которых является A', отвечает сам элемент A', и обратно.

Возьмем в классе $\bar{A}$, состоящем из элементов, имеющих образ A', два элемента A_1 и A_2, и в классе $\bar{B}$, состоящем из элементов с образом B', — два элемента B_1 и B_2. Поскольку φ — гомоморфизм, имеем

$$\varphi(A_1B_1) = \varphi(A_1)\varphi(B_1) = A'B' = \\ = \varphi(A_2)\varphi(B_2) = \varphi(A_2B_2).$$

Результаты композиции A_1B_1 и A_2B_2 имеют один и тот же образ и, следовательно, принадлежат одному и тому же классу относительно $\mathfrak{R}$. Класс, которому принадлежит результат композиции двух элементов, зависит только от классов этих элементов. Итак, доказана

Теорема. *Всякому гомоморфизму* φ *полугруппы с единицей* $\mathbf{M}$ *на или в другую полугруппу с единицей можно поставить в соответствие конгруэнцию* $\mathfrak{R}$, *определяемую совпадением образов.*

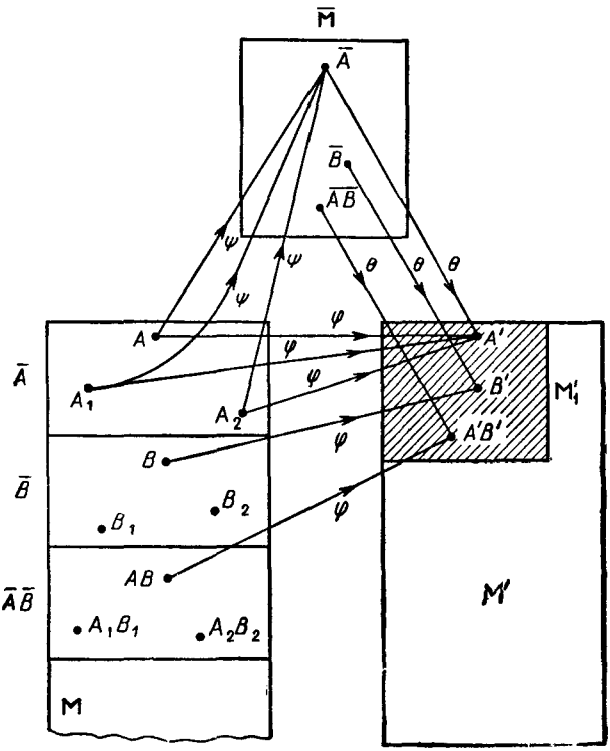
Конгруэнция $\mathfrak{R}$ приводит нас к ситуации, описанной в п. 13.1.2: существует естественный гомоморфизм $\mathbf{M}$ на $\mathbf{M}/\mathfrak{R} = \overline{\mathbf{M}}$, где $\overline{\mathbf{M}}$ — также полугруппа с единицей.

Взаимно однозначное соответствие θ между $\overline{\mathbf{M}}$ и $\mathbf{M}'_1$ (множеством образов) перестановочно с операциями композиции:

$$\left[A' \underset{\theta^{-1}}{\overset{\theta}{\rightleftarrows}} \overline{A}\ \&\ B' \underset{\theta^{-1}}{\overset{\theta}{\rightleftarrows}} \overline{B}\right] = \left[A'B' \underset{\theta^{-1}}{\overset{\theta}{\rightleftarrows}} \overline{A}\,\overline{B}\right].$$

Таким образом, θ является изоморфизмом.

В итоге мы получаем следующую схему:



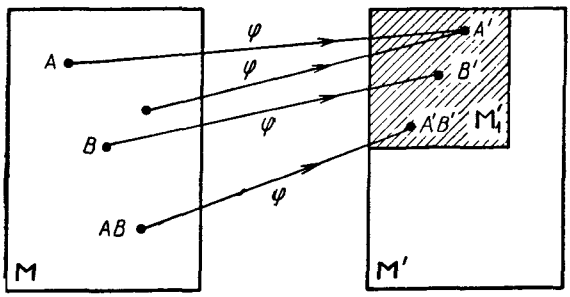
13.1.4. Классификация гомоморфизмов

Сохраним здесь обозначения, введенные в предыдущем пункте.

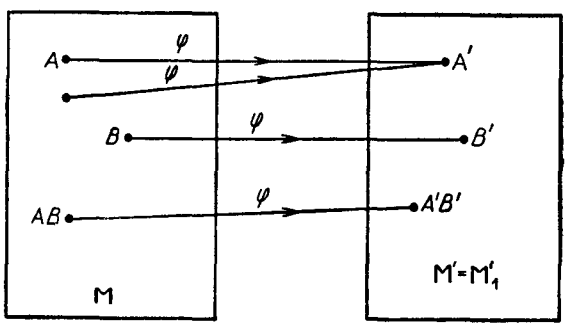
Рассмотрим отображение φ более подробно; имеются следующие возможности:

1) φ *не сюръективно.* Если существуют элементы множества $\mathbf{M}'$, не являющиеся образами никаких элементов из $\mathbf{M}$, то мы имеем

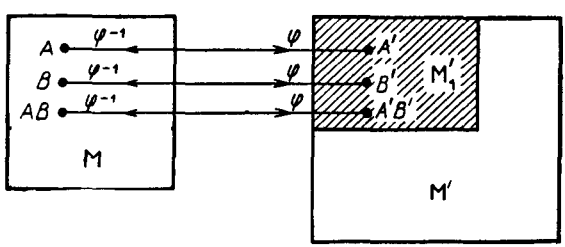
строгое включение: $\mathbf{M}'_1 \subset \mathbf{M}'$, $\mathbf{M}'_1 \neq \mathbf{M}_1$. Тогда о φ следует говорить как о *гомоморфизме множества* $\mathbf{M}$ *строго в* $\mathbf{M}'$. В качестве иллюстрации этой ситуации см. следующий рисунок:



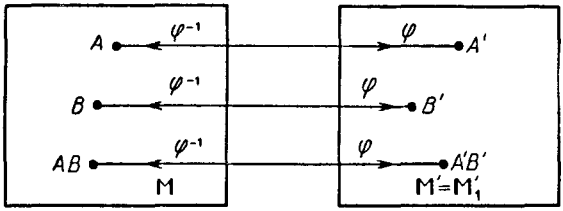
2) φ *сюръективно*. Если всякий элемент $\mathbf{M}'$ есть образ какого-либо элемента $\mathbf{M}$, то $\mathbf{M}' = \mathbf{M}'_1$, и φ есть *гомоморфизм* $\mathbf{M}$ *на* $\mathbf{M}'$ или, короче, *эпиморфизм*:



3) φ *инъективно*. Если образ A' всякого элемента A является образом только этого элемента, говорят, что φ — *мономорфизм* $\mathbf{M}$ *в* $\mathbf{M}'$:



4) φ *биективно* (взаимно однозначно). Если всякий элемент множества $\mathbf{M}'$ является образом одного и только одного элемента множества $\mathbf{M}$, говорят, что φ — *изоморфизм* $\mathbf{M}$ *на* $\mathbf{M}'$:



Примеры. 1. $\mathbf{M}$ — свободная полугруппа над алфавитом $\mathfrak{A}$ и $\mathbf{M}'$ — полугруппа (в данном случае — группа), состоящая из двух элементов $\{p, i\}$ и определяемая следующей таблицей:

+	p	i
p	p	i
i	i	p

Отображая цепочки четной длины на p, а цепочки нечетной длины на i, мы получим эпиморфизм.

2. $\mathbf{M}$ — полугруппа целых неотрицательных чисел относительно сложения; $\mathbf{M}'$ — свободная полугруппа над $\{a, b\}$. Отображая нуль в пустую цепочку, 1 — в a, 2 — в aa, n — в a^n, мы получим мономорфизм.

Случай совпадения $\mathbf{M}$ с $\mathbf{M}'$. Когда гомоморфизм φ отображает полугруппу $\mathbf{M}$ в (возможно, на) себя, φ называется *эндоморфизмом*; если при этом φ — изоморфизм, он называется *автоморфизмом*.

Однако взаимно однозначное отображение множества $\mathbf{M}$ на его собственное подмножество, перестановочное с композицией, называют по-прежнему мономорфизмом. Заметим сразу же, что такие мономорфизмы существуют.

Пример. $\mathbf{M}$ — свободная полугруппа с единицей над $\{a, b\}$. Каждой цепочке ставится в соответствие цепочка, получаемая удвоением каждой буквы: $a \to aa$, $ab \to aabb$, $abb \to aabbbb$ и т. д. Это отображение взаимно однозначно и перестановочно с конкатенацией.

13.1.5. Ядро гомоморфизма

Будем считать, что полугруппы $\mathbf{M}$ и $\mathbf{M}'$ обладают каждая единичным элементом E и E' соответственно (быть может, после добавления такового). Пусть φ — гомоморфизм множества $\mathbf{M}$ в (возможно, на) $\mathbf{M}'$; положим $\varphi(E) = E_1$.

Тогда

$$\varphi(EA) = \varphi(E)\varphi(A) = E_1\varphi(A);$$
$$\varphi(AE) = \varphi(A)\varphi(E) = \varphi(A)E_1,$$

Далее, $EA = AE = A$ (поскольку E — единичный элемент); следовательно,

$$E_1\varphi(A) = \varphi(A)E_1 = \varphi(A).$$

Таким образом, E_1 является единицей для полугруппы $\mathbf{M}_1$ — множества образов; однако из этого без дополнительных предположений еще не следует, что $E_1 = E'$, т. е. что E_1 — единица полугруппы $\mathbf{M}'$.

Можно утверждать, что $E_1 = E'$, если известно, что $E' \in \mathbf{M}_1$ (в частности, если гомоморфизм является эпиморфизмом). В самом деле, в $\mathbf{M}_1'$ имеем

$E'E_1 = E_1$ в силу того, что E' — единица;

$E'E_1 = E'$ в силу того, что E_1 — единица; отсюда $E_1 = E'$.

В дальнейшем всюду, если не оговорено противное, мы будем считать, что $E_1 = E'$.

В теории групп *ядром* гомоморфизма φ, отображающего группу $\mathbf{G}$ в (возможно, на) группу $\mathbf{G}'$, называют множество $\mathbf{H}$ элементов группы $\mathbf{G}$, имеющих в качестве образа единичный элемент группы $\mathbf{G}'$. Классический результат теории групп гласит, что ядро является подгруппой группы $\mathbf{G}'$ и его задание полностью определяет гомоморфизм.

Для полугрупп с единицей положение оказывается более сложным: здесь, вообще говоря, восстановить гомоморфизм по одному классу канонической конгруэнции невозможно.

Ничто не мешает, впрочем, и здесь называть *ядром гомоморфизма* тот класс, к которому принадлежит единичный элемент исходной полугруппы. В рамках интересующей нас проблематики понятие ядра гомоморфизма занимает важное место (хотя и не столь важное, как в теории групп); в дальнейшем мы будем неоднократно пользоваться им.

Более или менее существенная роль ядра в том или ином конкретном случае зависит от роли единичного элемента в полугруппе образов, т. е. в конечном счете решающая роль принадлежит образам.

§ 13.2. КОНГРУЭНЦИЯ И ЭКВИВАЛЕНТНОСТИ, СОПОСТАВЛЯЕМЫЕ ЯЗЫКУ

13.2.1. Конгруэнция по допустимым контекстам (взаимозамещаемость)

Пусть $\mathfrak{A}$ — алфавит, $\mathfrak{A}^*$ — свободная полугруппа над $\mathfrak{A}$, $L \subset \mathfrak{A}^*$ — язык и $M \in \mathfrak{A}^*$ — некоторая цепочка. Если существуют цепочки,

$A, B \in \mathfrak{A}^*$, такие, что $AMB \in L$, мы будем называть упорядоченную пару (A, B) *допустимым контекстом цепочки* M *относительно* L.

Примеры. Пусть $\mathfrak{A} = \{a, b, c\}$, L — «зеркальный язык» $\{X c \tilde{X} \mid X \in \{a, b\}^*\}$. Для цепочки ac пара (b, ab) есть допустимый контекст относительно L. Множество допустимых контекстов для ac состоит из всех пар вида $(X, a\tilde{X})$.

Для цепочки aca множество допустимых контекстов есть $\{X, \tilde{X} \mid X \in \{a, b\}^*\}$; оно содержит, в частности, пару (E, E), где E — пустая цепочка.

Для цепочки $accb$ нет ни одного допустимого контекста относительно L, т. е. множество допустимых контекстов этой цепочки пусто.

NB: заметим, что если множество допустимых контекстов включает «пустой контекст», т. е. пару (E, E), то тем самым оно само не пусто!

Определим на свободной полугруппе $\mathfrak{A}^*$ отношение $\equiv_L$ следующим образом. Будем писать

$$M \equiv_L P,$$

если множество допустимых контекстов цепочки M относительно языка L совпадает с множеством допустимых контекстов цепочки P относительно того же языка.

Другими словами, $\equiv_L$ означает, что

— либо ни для M, ни для P нет допустимых контекстов относительно L;

— либо и для M, и для P имеются допустимые контексты относительно L, причем всякая пара, являющаяся допустимым контекстом для M, является допустимым контекстом и для P, и наоборот.

В более компактной форме это утверждение записывается так:

$$[M \equiv_L P] \Leftrightarrow (\forall X \in \mathfrak{A}^*)(\forall Y \in \mathfrak{A}^*)[XMY \in L \Leftrightarrow XPY \in L].$$

Отношение $\equiv_L$ называется *взаимозамещаемостью* относительно L.

Пример. Если $\mathfrak{A} = \{a, b\}$ и L — конечный язык $\{aaa, aba\}$, то $aaa \equiv_L aba$, причем соответствующее множество допустимых контекстов есть $\{(E, E)\}$, и $bab \equiv_L bbb$, причем соответствующее множество допустимых контекстов пусто.

Ясно, что $\equiv_L$ есть отношение эквивалентности. Посмотрим, как оно ведет себя относительно конкатенации.

Пусть A и B — произвольные цепочки и $P \equiv_L M$; сравним цепочки AMB и APB. Любому контексту (X, Y), независимо от того, является ли он допустимым для M и P, можно сопоставить контекст (XA, BY). Тогда мы будем иметь

$$X(AMB)Y \in L \Leftrightarrow (XA)M(BY) \in L \Leftrightarrow (XA)P(BY) \in L \Leftrightarrow$$
$$\Leftrightarrow X(APB)Y \in L.$$

Таким образом, всякий контекст, допустимый для AMB, является допустимым и для APB; ясно, что обратное также верно. Поэтому

$$P \equiv_L M \Rightarrow APB \equiv_L AMB.$$

Следовательно, отношение $\equiv_L$ (взаимозамещаемость) является конгруэнцией. Мы будем называть его еще *конгруэнцией по допустимым контекстам относительно* L.

Пример.

$$\mathfrak{A} = \{a, b\}; \quad L = \{a^m b^n a^p \mid m,\ n \geqslant 1,\ p \geqslant 0\}.$$

Цепочки, не принадлежащие L, либо имеют вид a^m, либо имеют начальный сегмент вида b^r, $r \geqslant 1$, или вида $a^m b^n a^p b^r$, $m, n, p, r \geqslant 1$.

Все эти цепочки принадлежат одному классу, который соответствует пустому множеству допустимых контекстов.

Цепочки языка L распределяются по двум классам: класс цепочек вида $a^m b^n$, $m, n \geqslant 1$, имеющих допустимые контексты вида $(a^x, b^y a^z \mid x, y, z \geqslant 0)$, и класс цепочек вида $a^m b^n a^p$, $p \geqslant 1$, имеющих допустимые контексты вида (a^x, a^z). Таким образом, L есть объединение этих двух классов.

Отношение $\equiv_L$ задает разбиение множества $\mathfrak{A}^*$ на непересекающиеся классы и определяет фактормножество $\mathfrak{A}^*/\equiv_L$, являющееся полугруппой (с единицей) относительно операции, индуцированной конкатенацией.

Если $M \in L$ и $M \equiv_L P$, то контекст (E, E), допустимый для M, является допустимым и для P, откуда $P \in L$. Если L содержит M, то L содержит и весь класс цепочек, находящихся с M в отношении $\equiv_L$.

Теорема. *Язык L может быть представлен как объединение некоторых классов взаимозамещаемых цепочек.*

Ср. предыдущий пример.

13.2.2. Основное свойство взаимозамещаемости

Докажем следующую теорему.

Теорема. *Всякая конгруэнция, определенная на $\mathfrak{A}^*$ и такая, что L является объединением некоторых ее классов, либо тоньше*[1]), *чем конгруэнция $\equiv_L$, либо совпадает с $\equiv_L$.*

Пусть $\mathfrak{R}$ — такая конгруэнция. Покажем, что из $A\mathfrak{R}B$ следует $A \equiv_L B$. В самом деле,

$$[A\mathfrak{R}B \,\&\, XAY \in L] \Rightarrow [XAY\mathfrak{R}XBY \,\&\, XAY \in L];$$

а поскольку язык L есть объединение $\mathfrak{R}$-классов, он содержит XBY, если только он содержит XAY.

[1]) Эквивалентность $\sim_1$ *тоньше* эквивалентности $\sim_2$, если всякий $\sim_1$-класс содержится в некотором $\sim_2$-классе, и при этом $\sim_1$ и $\sim_2$ не совпадают. — *Прим ред.*

Таким образом, любой контекст, допустимый для A, допустим и для B, что и требовалось доказать.

13.2.3. Эквивалентность по допустимым концам

Если ограничиться контекстами, в которые левый (соответственно правый) член есть пустая цепочка, то из отношения $\equiv_L$ мы получим другое отношение — *эквивалентность по допустимым концам* (соответственно *началам*).

Поскольку это понятие важно для нас, мы рассмотрим его независимо от взаимозамещаемости.

Пусть L — язык, содержащийся в свободной полугруппе $\mathfrak{A}^*$, а M и X — цепочки, принадлежащие $\mathfrak{A}^*$. Если $MX \in L$, мы будем называть X *допустимым концом для* M *относительно* L.

Пример. $\mathfrak{A} = \{a, b, c\}$, L — «зеркальный язык»:

$$L = \{AcA^{\sim} \mid A \in \{a, b\}^*\}.$$

Тогда

для *aabc* существует ровно один допустимый конец, а именно *baa*;

для *aab* допустимым концом является любая цепочка вида $Xc\tilde{X}baa$;

для *aabcbaa* существует ровно один допустимый конец — пустая цепочка;

для *aabcbb* ни одна цепочка не является допустимым концом (множество допустимых концов для этой цепочки пусто).

Если пустая цепочка E является допустимым концом для A, то множество допустимых концов для A не пусто.

Определим теперь на множестве $\mathfrak{A}^*$ отношение $\Im_L$ следующим образом: $A\Im_L B$ тогда и только тогда, когда A и B имеют одни и те же допустимые концы относительно L.

Это отношение, являющееся, разумеется, эквивалентностью, имеет место между A и B в следующих двух случаях:

1) либо ни A, ни B не имеют допустимых концов;

2) либо и для A, и для B существуют допустимые концы, причем всякая цепочка, являющаяся допустимым концом для A, является допустимым концом для B, и обратно.

Пусть $A\Im_L B$ и $C \in \mathfrak{A}^*$ — произвольная цепочка, рассмотрим цепочки AC и BC.

Если множество допустимых концов для A и аналогичное множество для B пусты, то множества допустимых концов для AC и BC также пусты.

Если X — допустимый конец для AC, то $ACX \in L$ и, следовательно, CX — допустимый конец для A; тогда CX — допустимый конец и для B. Поэтому $BCX \in L$, откуда следует, что X — допустимый конец и для BC.

Следовательно, отношение $\mathfrak{I}_L$ совместимо вправо с конкатенацией.

Рассматривая допустимые начала цепочек, можно ввести аналогичное отношение $\mathfrak{T}_L$, совместимое с конкатенацией влево.

Нетрудно видеть, что L есть объединение некоторых классов по отношению $\mathfrak{I}_L$. В самом деле, из $M \in L$ следует, что множество допустимых концов для M содержит пустую цепочку E. Следовательно, для всякой цепочки P, находящейся с M в отношении $\mathfrak{I}_L$, пустая цепочка E также является допустимым концом, откуда $PE \in L$, т. е. $P \in L$. Таким образом, если L содержит некоторую цепочку, то L содержит весь класс, в который входит эта цепочка.

Все сказанное можно сформулировать в виде следующего предложения.

Предложение. *Язык $L \subset \mathfrak{A}^*$ определяет в $\mathfrak{A}^*$ отношение эквивалентности $\mathfrak{I}_L$ по допустимым концам (соответственно отношение эквивалентности $\mathfrak{T}_L$ по допустимым началам), совместимое вправо (соответственно влево) с конкатенацией.*

Сам язык L есть объединение некоторых классов эквивалентности по $\mathfrak{I}_L$ (соответственно по $\mathfrak{T}_L$).

13.2.4. Свойства отношения $\mathfrak{I}_L$

Отношение $\mathfrak{I}_L$ обладает следующими двумя свойствами:

1. Оно совместимо вправо с конкатенацией.
2. L является объединением классов эквивалентности по этому отношению.

Нетрудно видеть, что всякое отношение $\mathfrak{R}$, обладающее этими двумя свойствами, либо тоньше, чем $\mathfrak{I}_L$, либо совпадает с ним.

Действительно, пусть $\mathfrak{R}$ — такое отношение; покажем, что из $A\mathfrak{R}B$ следует $A\mathfrak{I}_L B$. В самом деле,

$$[A\mathfrak{R}B \,\&\, AC \in L] \Rightarrow [AC\mathfrak{R}BC \,\&\, AC \in L]$$

(в силу совместимости $\mathfrak{R}$ с конкатенацией вправо); но

$$[AC\mathfrak{R}BC \,\&\, AC \in L] \Rightarrow BC \in L,$$

так как L, будучи объединением классов эквивалентности, содержит BC, если только L содержит AC.

Таким образом, всякий допустимый конец для A допустим и для любого B, такого, что $A\mathfrak{R}B$, и обратно, что и требовалось доказать.

Итак, доказана

Теорема. *Отношение эквивалентности по допустимым концам относительно языка L является не более тонким, чем любое отношение эквивалентности, совместимое вправо с конкатенацией и такое, что L есть объединение классов по этому отношению.*

§ 13.3. ПОНЯТИЯ, СВЯЗАННЫЕ С КОДАМИ

13.3.1. Определение кода

Пусть X и Y — конечные алфавиты, состоящие не менее чем из двух букв каждый; тогда всякий мономорфизм множества Y^* в X^* называется *кодированием*.

Как всякий гомоморфизм, мономорфизм определяется образами образующих полугруппы Y^*, т. е. образами букв алфавита Y. Множество образов букв алфавита Y называется *кодом*; кодирование становится возможным благодаря наличию кода.

Пример. Пусть $Y = \{x, y, z, t\}$, $X = \{0, 1\}$; в качестве образов букв x, y, z, t возьмем соответственно 00, 01, 10, 11. Ясно, что мы определили мономорфизм Y^* в X^*. Если цепочка из X^* является образом некоторой цепочки из Y^*, то она является образом только этой цепочки.

Например:

11 01 01 01 00 расшифровывается как $t\,y\,y\,y\,x$.

Задавая образы букв алфавита Y случайным образом, мы получим гомоморфизм, который не обязательно будет мономорфизмом.

Пример. Сохранив алфавиты предыдущего примера, зададим образы букв так: $x \to 0$, $y \to 1$, $z \to 10$ и $t \to 11$ (т. е. первые четыре натуральных числа в двоичной записи). Тогда, например, 10 является одновременно образом буквы z и цепочки yx: расшифровка уже не является однозначной. Таким образом, $\{00, 01, 10, 11\}$ — код, а $\{0, 1, 01, 11\}$ — не код.

Задачей алгебраической теории кодирования является характеризация тех подмножеств множества X^*, которые являются кодами, т. е. тех подмножеств, которые порождают свободную подполугруппу.

Заметим, что если A — код, то полугруппу A^* называют *множеством сообщений*.

13.3.2. Необходимое и достаточное условие кодовости

Для того чтобы $A \subset X^*$ было кодом, необходимо и достаточно, чтобы выполнялось следующее условие:

$$(L)\quad (\forall f \in X^*)[(A^* f \cap A^* \neq \emptyset \,\&\, fA^* \cap A^* \neq \emptyset) \Rightarrow f \in A^*)].$$

Доказательство. Вместо утверждения $(L) \Leftrightarrow$ (A есть код) мы докажем равносильное утверждение

$$(A \text{ не есть код}) \Leftrightarrow \neg(L).$$

1. (A не есть код) $\Rightarrow \neg(L)$.

Если $A = \{a_1, \ldots, a_s\}$ — не код, то множество цепочек из A^*, допускающих каждая по меньшей мере два чтения, не пусто, и в этом

множестве можно выделить подмножество цепочек минимальной длины. Пусть

$$m = a_{i_1} \dots a_{i_n} = a_{j_1} \dots a_{j_p} \tag{1}$$

— одна из таких цепочек. Минимальность длины m влечет за собой неравенство $a_{i_1} \neq a_{j_1}$. Не уменьшая общности изложения, можно предположить, что

$$|a_{i_1}| < |a_{j_1}|,$$

откуда

$$a_{j_1} = a_{i_1} h, \quad h \in X^*, \quad |h| \neq 0. \tag{2}$$

Из (1) и (2) следует, что

$$a_{i_2} \dots a_{i_n} = h a_{j_2} \dots a_{j_p}. \tag{3}$$

Из (2) вытекает, что

$$A^* h \cap A^* \neq \emptyset,$$

а соотношение (3) влечет за собой

$$h A^* \cap A^* \neq \emptyset;$$

однако из (3) и минимальности m следует, что $h \notin A^*$. Таким образом, (L) не имеет места: из допущения, что A — не код, следует «не (L)».

2. $\neg(L) \Rightarrow (A$ не есть код$)$.

По предположению найдутся такие

$$f \in X^*, \quad g \in A^*, \quad h \in A^*, \quad g' \in A^*, \quad h' \in A^*,$$

что

$$gf = h, \quad fg' = h', \quad f \notin A^*.$$

Из условия $f \notin A^*$ следует, что f не пусто; следовательно, и h не пусто и, во всяком случае, $g \neq h$. Но из соотношений $gfg' = hg' = gh'$, $g \neq h$ следует, что A не является кодом.

Глава XIV

ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ ОБ АВТОМАТНЫХ ЯЗЫКАХ

§ 14.1. СТАНДАРТНЫЕ А-ЯЗЫКИ

14.1.1. Определение

Пусть V_T — терминальный алфавит. Определим язык K_S следующими условиями:

. все цепочки языка K_S начинаются с некоторой буквы, принадлежащей выделенному подмножеству I алфавита V_T (I — множество допустимых начальных букв);

.. все цепочки языка K_S оканчиваются некоторой буквой, принадлежащей другому выделенному подмножеству J алфавита V_T (J — множество допустимых конечных букв);

... в цепочках языка K_S не встречаются диграммы (пары стоящих рядом букв), входящие в заданное множество $\Delta \subset V_T \times V_T$.

Пример.

$$V_T = \{a, b, c, d, f\};\ I = \{a, b, c\};\ J = \{c, f\}.$$

Множество Δ запрещенных диграмм мы представим с помощью матрицы следующего вида:

		Конечные буквы				
	Первая буква \ Вторая буква	*a*	*b*	*c*	*f*	*d*
Начальные буквы	*a*	0	0	1	0	1
	b	0	1	1	1	0
	c	0	0	0	0	1
	f	0	0	1	1	1
	d	1	1	1	1	0

Стоящий на пересечении строки x и столбца y элемент 0 означает, что диаграмма xy запрещена, 1 — что она разрешена.

Языки, удовлетворяющие сформулированным условиям, являются А-языками; мы будем называть такие А-языки *стандартными*.

В самом деле, множество тех цепочек в V_T, которые начинаются буквой из I и оканчиваются буквой из J, может быть представлено в виде

$$IV_T^* \cap V_T^* J,$$

а множество цепочек, содержащих запрещенные диграммы, — в виде

$$V_T^* \Delta V_T^*.$$

Следовательно, язык K_S представляется как пересечение:

$$K_S = [IV_T^* \cap V_T^* J] \cap [V_T^* \setminus V_T^* \Delta V_T^*].$$

Поскольку V_T^*, I, J, Δ суть А-языки (I, J, Δ конечны), из результатов § 10.4 следует, что K_S есть А-язык.

14.1.2. Система уравнений, определяющая язык K_S

Пусть имеется язык K_S, заданный четверкой

$$(V_T, I, J, \Delta),$$

где

$$V_T = \{a_\lambda \mid 1 \leqslant \lambda \leqslant n\}, \quad I = \{a_{\alpha_1}, \ldots, a_{\alpha_i}\}, \quad J = \{a_{\beta_1}, \ldots, a_{\beta_j}\}.$$

Для случая, когда некоторые буквы могут быть и начальными и конечными, положим

$$I \cap J = \{a_{\gamma_1}, \ldots, a_{\gamma_k}\}.$$

Множество Δ запрещенных диграмм мы зададим с помощью функции $\delta(\lambda, \mu)$, такой, что

$$\delta(\lambda, \mu) = \begin{cases} 0, & \text{если диграмма } a_\lambda a_\mu \text{ запрещена,} \\ 1, & \text{если диграмма } a_\lambda a_\mu \text{ разрешена.} \end{cases}$$

Эта функция фактически использовалась в предыдущем пункте: ею определяются элементы матрицы запрещенных диграмм.

Построим формальный степенной ряд S, опорным множеством которого будет язык K_S. Введем сначала вспомогательные ряды $A_1, \ldots, A_n$, где n равно числу терминальных символов $a_1, \ldots, a_n$.

Говоря более точно, букве $a_\lambda \in V_T$ ставится в соответствие (некоммутативный) формальный степенной ряд A_λ, такой, что

1) все цепочки опорного множества ряда A_λ можно присоединять конкатенацией справа к a_λ, т. е. если некоторая цепочка ряда A_λ начинается буквой a_μ, то диаграмма $a_\lambda a_\mu$ разрешена: $\delta(\lambda, \mu) = 1$;

2) цепочки опорного множества ряда A_λ содержат только разрешенные диграммы;

3) всякая цепочка опорного множества ряда A_λ оканчивается буквой из J.

Построим теперь систему из $n+1$ уравнений; эти уравнения будут иметь в качестве первых членов $S, A_1, \ldots, A_n$ соответственно. Первое уравнение имеет вид

$$S = a_{\alpha_1}A_{\alpha_1} + \ldots + a_{\alpha_i}A_{\alpha_i} + a_{\gamma_1} + \ldots + a_{\gamma_k};$$

оно гарантирует нам, что цепочки языка K_S будут начинаться с допустимых букв и что K_S будет содержать все однобуквенные цепочки из $I \cap J$.

Остальные n уравнений построены по следующей общей схеме:

$$A_\lambda = \sum_{\mu=1}^{l} \delta(\lambda, \beta_\mu)\, a_{\beta_\mu} + \sum_{\mu=1}^{n} \delta(\lambda, \mu)\, a_\mu A_\mu; \qquad \lambda = 1, \ldots, n.$$

Здесь вторая сумма задает все буквы a_μ, присоединимые к a_λ конкатенацией справа, но не являющиеся концевыми в цепочке; первая сумма задает те из допустимых заключительных букв, которые могут присоединяться к a_λ конкатенацией справа.

Эта система уравнений порождает язык K_S (и вспомогательные языки, выводимые из $A_1, \ldots, A_n$ как из аксиом) и притом однозначно. Выписанная система соответствует, очевидно, правосторонней А-грамматике; это дает еще одно доказательство автоматности языка K_S.

Пример. Возьмем язык K_S, рассматривавшийся в примере на стр. 214. Его терминальный алфавит состоит из пяти букв: a, b, c, d, f; нетерминальный алфавит будет включать шесть букв: аксиому S и пять вспомогательных символов A, B, C, D, F.

Соответствующая система уравнений может быть записана в виде матрицы, имеющей 6 строк (по числу уравнений) и 7 столбцов: пять для одночленов типа aA, bB, ... и два — для заключительных букв c и f. Вот эта матрица:

	Допустимые начала	Допустимые концы
$S=$	$aA + bB + cC$	$+c$
$A=$	$cC \quad +dD$	$+c$
$B=$	$bB + cC + fF$	$+c+f$
$C=$	dD	
$F=$	$cC + fF + dD$	$+c+f$
$D=$	$aA + bB + cC + fF$	$+c+f$

Из такой записи системы уравнений можно непосредственно «вычитать» множество допустимых начал, множество допустимых концов и матрицу переходов, т. е. матрицу разрешенных диграмм (клетки, взятые в полужирную рамку).

14.1.3. Упражнения

1. Исследовать стандартный А-язык, определяемый четверкой (V_T, I, J, Δ), где

$$V_T = \{a, b, c\}, \quad I = \{a, b\}, \quad J = \{c\}, \quad \Delta = \{bc, ca, cb, cc\}.$$

Написать соответствующую систему уравнений и решить ее; дать простую характеристику языка.

2. Указать достаточные условия того, чтобы четверка (V_T, I, J, Δ) определяла пустой язык, например:

а) «плохой выбор» множества I (построить пример);

б) «плохой выбор» множества J (построить пример).

§ 14.2. СВОЙСТВА СТАНДАРТНЫХ А-ЯЗЫКОВ

14.2.1. Теорема

Итак, всякий стандартный А-язык порождается системой «праволинейных» уравнений, определенной над терминальным алфавитом $V_T = \{a_\mu \mid 1 \leqslant \mu \leqslant n\}$ и вспомогательным алфавитом $V_A = \{A_\lambda \mid 0 \leqslant \lambda \leqslant n\}$ и удовлетворяющей следующим двум условиям (KS):

. нетерминальный символ A_0 (аксиома языка) встречается только в одном уравнении, и притом в его левой части;

.. все члены правых частей уравнений имеют вид $a_\mu A_\mu$ или a_μ.

Обратно, рассмотрим такую систему уравнений. Если записать ее в виде матрицы, рассмотренной в предыдущем пункте, то мы сможем сразу усмотреть в ней множество допустимых начал, множество допустимых концов и матрицу допустимых диграмм: это значит, что наша система задает стандартный А-язык.

Теорема. *Для того чтобы язык был стандартным А-языком, необходимо и достаточно, чтобы его можно было задать системой уравнений, удовлетворяющей условиям (KS).*

14.2.2. Стандартные А-языки и автоматы

Легко показать (мы предоставляем это читателю), что конечный автомат, сопоставляемый стандартному А-языку посредством системы уравнений типа (KS), является детерминированным автоматом.

Поскольку при этом запрещенные переходы задаются списком запрещенных диграмм, то для проверки допустимости предъявленной цепочки достаточно хранить в памяти автомата *одну* букву.

Тем самым автомат оказывается 1-определенным (см. п. 10.3.3); таким образом, *стандартный А-язык является 1-определенным.*

14.2.3. Отображения стандартных А-языков на произвольные А-языки

Стандартные А-языки интересны не только в силу своей особой простоты, но еще и потому, что с их помощью можно дать простое описание структуры произвольного А-языка. В самом деле, имеет место следующее утверждение.

Теорема. *Для всякого А-языка M существуют стандартный А-язык L и гомоморфизм φ, такие, что $\varphi(L) = M$.*

Доказательство. Пусть язык M порождается А-грамматикой G с терминальным алфавитом V_T. Возьмем в качестве нового терминального алфавита V_T' множество всевозможных пар вида (a, A), где $a \in V_T$ и A — вершина графа грамматики, и определим стандартный А-язык L в алфавите V_T следующим образом:

— допустимыми началами будут пары (a, A), такие, что в графе G имеется дуга, ведущая из начальной вершины в A и помеченная символом a;

— допустимыми концами будут пары (b, B), такие, что B — заключительная вершина и b — пометка на некоторой дуге, ведущей в b;

— разрешенными диграммами будут пары $((a_1, A_1), (a_2, A_2))$, такие, что: (а) a_1 — пометка на некоторой дуге, ведущей в A_1; (б) имеется дуга, ведущая из A_1 в A_2 и помеченная символом a_2.

Попросту говоря, L будет состоять из всех «полных путей» в графе грамматики G.

Гомоморфизм φ, определенный условием $\varphi(a, A) = a$, будет, очевидно, отображать L на M.

Замечание. Эту теорему нетрудно было бы доказать и чисто алгебраическим методом, построив систему уравнений, определяющую L, по системе, определяющей M.

§ 14.3. АЛГЕБРАИЧЕСКОЕ ОПИСАНИЕ А-ЯЗЫКОВ

Поскольку один и тот же А-язык может порождаться несколькими разными А-грамматиками или конечными автоматами, описание А-языка с помощью конкретной грамматики или конкретного автомата может обладать «случайными» особенностями, которые не являются «внутренне» присущими языку. Чтобы избавиться от подобных особенностей, целесообразно попытаться дать чисто алгебраическое описание А-языков, трактуя их как подмножества свободной полугруппы.

Из методических соображений мы начнем с рассмотрения некоторого конечного автомата, от которого в ходе рассуждений мы сможем затем избавиться.

14.3.1. Продолжение функции переходов

Рассмотрим следующий (детерминированный) конечный автомат:

. алфавит $\mathfrak{A} = \{a_i \mid 1 \leqslant i \leqslant n\}$;

.. множество состояний $\Sigma = \{S_j \mid 0 \leqslant j \leqslant p\}$;

... начальное состояние $S_0 \in \Sigma$ и множество заключительных состояний $\Sigma_\omega \subset \Sigma$;

.... функция переходов f, заданная множеством команд вида $(a_i, S_j) \to S_k$.

Функция переходов определена на некотором подмножестве декартова произведения $\mathfrak{A} \times \Sigma$; ситуации, принадлежащие дополнению к этому подмножеству, не появляются в левых частях команд автомата, и в случае, если автомат оказывается в подобной ситуации, он останавливается. Присоединим к множеству состояний Σ *состояние остановки* \$ и положим $f(a_i, S_j) = \$$ для любой пары (a_i, S_j), не являющейся левой частью какой-либо команды. Теперь функция переходов определена на всем множестве $\mathfrak{A} \times \Sigma$ и принимает значения из $\Sigma \cup \{\$\}$.

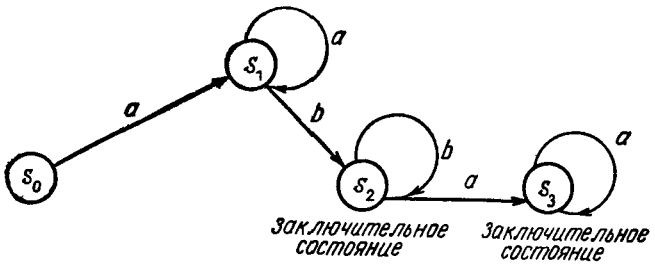
Продолжим далее эту функцию переходов f на множество $\mathfrak{A}^* \times \Sigma$ следующим образом (ср. п. 10.2.3): пусть X — цепочка из $\mathfrak{A}^*$ и S_j — состояние автомата; тогда значением функции f_g — продолженной функции переходов — для пары (X, S_j) будет состояние, в которое перейдет автомат после того, как, начав работать в состоянии S_j, он прочел (или пытался прочесть) слева направо цепочку X. Формальное определение f_g выглядит так: если $|X| = 1$, т. е. $X = a_i$, то $f_g(M, S_j) = f(a_i, S_j)$; если f_g определена для всех цепочек длины q и $X = b_1 \ldots b_q\, b_{q+1}$, $b_i \in \mathfrak{A}$, то $f_g(X, S_j) = f(b_{q+1}, f_g(b_1 \ldots b_q, S_j))$, если $f_g(b_1 \ldots b_q, S_j) \neq \$$, и $f_g(X, S_j) = \$$, если $f_g(b_1 \ldots b_q, S_j) = \$$.

Аналогичным образом можно определить функцию f_d, соответствующую чтению цепочек справа налево.

Пример. Пусть имеется автомат: $\mathfrak{A} = \{a, b\}$; $\Sigma = \{S_0, S_1, S_2, S_3\}$; начальное состояние S_0; $\Sigma_\omega = \{S_2, S_3\}$; f задана таблицей

	a	b
S_0	S_1	\$
S_1	S_1	S_2
S_2	S_3	S_2
S_3	S_3	\$

Этот автомат представим с помощью диаграммы (графа)



и допускает язык

$$\{a^m b^n a^q \mid 1 \leqslant m,\ 1 \leqslant n,\ 0 \leqslant q\}.$$

Имеем

$$f_g(abba, S_0) = S_3; \quad f_g(abab, S_0) = \$;$$
$$f_g(abba, S_1) = S_3; \quad f_g(abbb, S_1) = S_2 \text{ и т. д.}$$

14.3.2. Разбиение свободной полугруппы, определяемое конечным автоматом

Пусть имеется конечный автомат с алфавитом $\mathfrak{A}$; пусть, далее, f_g — функция переходов, соответствующая чтению цепочек множества $\mathfrak{A}^*$ слева направо, и S_i — состояние автомата. Рассмотрим отношение $\mathfrak{G}$ на $\mathfrak{A}^*$, определенное следующим образом:

$$A \mathfrak{G} B \Leftrightarrow (\forall S_i)[f_g(A, S_i) = f_g(B, S_i)].$$

Ясно, что $\mathfrak{G}$ есть отношение эквивалентности.

Пусть $A \mathfrak{G} B$ и C — произвольная цепочка из $\mathfrak{A}^*$. Тогда из равенств

$$f_g(A, S_i) = f_g(B, S_i) = S_k$$

следует, что

$$f_g(AC, S_i) = f_g(C, S_k) = f_g(BC, S_i).$$

Таким образом, отношение $\mathfrak{G}$ совместимо с конкатенацией вправо.

Классам разбиения, индуцируемого отношением $\mathfrak{G}$, взаимно однозначно соответствуют отображения Σ в $\Sigma \cup \{\$\}$ (Σ — множество состояний автомата), получаемые при фиксировании A в $f_g(A, S)$. Поэтому число классов не превосходит числа всевозможных таких отображений, которое равно $(n+1)^n$, где n — мощность Σ.

Следовательно, фактормножество $\mathfrak{A}^*/\mathfrak{G}$ конечно; иначе говоря, *индекс отношения* $\mathfrak{G}$ *конечен* [1]).

---

[1]) *Индексом* отношения эквивалентности $\mathfrak{R}$, определенного на множестве $\mathbf{M}$, называется число классов эквивалентности, на которые $\mathfrak{R}$ разбивает $\mathbf{M}$. — *Прим ред.*

Рассматриваемый автомат допускает цепочку X тогда и только тогда, когда он может ее прочесть, начав чтение в начальном состоянии S_0 и закончив в одном из заключительных состояний, иначе говоря, когда $f_g(X, S_0) \in \Sigma_\omega$, где Σ_ω — множество заключительных состояний. Это означает, что допускаемый автоматом язык представляет собой объединение некоторых классов разбиения, индуцированного отношением $\mathfrak{G}$, а именно тех классов, для которых соответствующее отображение Σ в $\Sigma \cup \{\$\}$ принимает на S_0 значения, принадлежащие Σ_ω.

Аналогичным образом можно было бы рассуждать, заменив f_g на f_d.

Итак, имеет место

Предложение. *Конечный автомат A с алфавитом $\mathfrak{A}$ определяет на свободной полугруппе $\mathfrak{A}^*$ отношение $\mathfrak{G}$ (соответственно $\mathfrak{P}$), имеющее конечный индекс, совместимое вправо (соответственно влево) с конкатенацией и такое, что язык, допускаемый автоматом A, есть объединение некоторых классов разбиения, индуцируемого отношением $\mathfrak{G}$ (соответственно $\mathfrak{P}$).*

Пример. Рассмотрим автомат, определенный в предыдущем пункте. Порождаемый им язык представляет собой объединение языка

$$\{a^m b^n \mid m \geqslant 1,\ n \geqslant 1\},\ \text{соответствующего состоянию } S_2,$$

и языка

$$\{a^p b^q a^r \mid p \geqslant 1,\ q \geqslant 1,\ r \geqslant 1\},\ \text{соответствующего состоянию } S_3.$$

Итак, мы показали, что для всякой свободной полугруппы существуют язык L и отношение эквивалентности $\mathfrak{R}$, такие, что

1. $\mathfrak{R}$ совместимо вправо (соответственно влево) с конкатенацией.
2. L представляет собой объединение классов эквивалентности по $\mathfrak{R}$.
3. $\mathfrak{R}$ имеет конечный индекс.

Мы знаем, однако, что отношение эквивалентности по допустимым концам относительно L (п. 13.2.3) обладает свойствами 1 и 2, а также что эта эквивалентность — наименее тонкая из всех эквивалентностей, обладающих этими свойствами (п. 13.2.4); поэтому индекс эквивалентности по допустимым концам — наименьший среди индексов эквивалентностей, обладающих указанными свойствами. Отсюда сразу вытекает

Следствие. *Отношение эквивалентности по допустимым концам относительно A-языка L имеет конечный индекс.*

14.3.3. Автомат с минимальным числом состояний

Докажем теперь обратное утверждение.

Допустим, что индекс отношения $\mathfrak{I}_L$ (эквивалентности по допустимым концам относительно L) конечен. Покажем, что в таком случае L обязательно является А-языком. Фактормножество $\mathfrak{A}^*/\mathfrak{I}_L$ конечно; построим автомат, обладающий тем свойством, что между его состояниями и классами, образующими множество $\mathfrak{A}^*/\mathfrak{I}_L$, можно установить взаимно однозначное соответствие.

Пусть $A \in \mathfrak{A}^*$; обозначим через $\bar{A}$ класс цепочки A и через $[\bar{A}]$ — состояние, отвечающее $\bar{A}$. Пустой цепочке E соответствует класс $\bar{E}$ и состояние $[\bar{E}]$, которое мы объявим начальным; цепочкам $M \in L$ будут соответствовать заключительные состояния.

Функция переходов f определяется условием

$$f(a_i, [\bar{M}]) = [\overline{Ma_i}].$$

Легко видеть, что все цепочки языка L, и только они, могут быть прочтены автоматом так, что в начале чтения он находится в состоянии $[\bar{E}]$, а в конце — в одном из состояний $[\bar{M}]$, таких, что $M \in L$; иначе говоря, автомат допускает язык L. Таким образом, L есть А-язык. Кроме того, поскольку число состояний построенного автомата равно индексу отношения $\mathfrak{I}_L$, этот автомат минимален [1]).

Итак, доказана

Теорема. *Если индекс отношения эквивалентности по допустимым концам конечен, то соответствующий язык является автоматным. Существует канонический способ построения такого автомата с минимальным числом состояний, который допускает данный язык.*

14.3.4. Конгруэнция, определяемая конечным автоматом

Пусть имеется конечный автомат с алфавитом $\mathfrak{A}$. Определим на $\mathfrak{A}^*$ отношение $\mathfrak{G}$, как в п. 14.3.2. Там же было показано, что это отношение является эквивалентностью, совместимой вправо с кон-

---

[1]) Минимальность понимается здесь в следующем смысле: никакой (детерминированный) конечный автомат, допускающий тот же язык L, не может иметь меньше состояний, чем данный. Чтобы убедиться в этом, достаточно показать, что, каковы бы ни были автомат A, допускающий язык L, и цепочки M и N, принадлежащие разным классам по отношению $\mathfrak{I}_L$, автомат A не может, начав чтение M и N в начальном состоянии S_0, заканчивать его в одном и том же состоянии. Но если бы чтение M и N, начатое в состоянии S_0, заканчивалось в одном и том же состоянии S_j, то (поскольку автомат в силу его детерминированности может прочесть цепочку, начиная с S_0, только одним способом) допустимыми концами и для M, и для N были бы те и только те цепочки, которые автомат может прочесть, начиная чтение в состоянии S_j и заканчивая его в одном из заключительных состояний, так что M и N принадлежали бы одному классу по $\mathfrak{I}_L$. — *Прим. ред.*

катенацией. Однако оно совместимо с конкатенацией также и влево, поскольку из $A\mathfrak{G}B$ следует, что

$$f_g(CA, S_i) = f_g(A, f(C, S_i)) = f_g(B, f(C, S_i)) = f_g(CB, S_i).$$

Таким образом, $\mathfrak{G}$ есть конгруэнция.

Если данный автомат допускает язык L, то отношение $\mathfrak{G}$ в силу теоремы п. 13.2.2 является не менее тонким, чем отношение $\equiv_L$. Однако число классов по $\mathfrak{G}$ конечно, и тем более это верно для $\equiv_L$. Итак, получена

Теорема. *Для всякого A-языка конгруэнция по допустимым контекстам имеет конечный индекс.*

14.3.5. Обратная теорема

Возникает следующая проблема.

Пусть L — такой язык, что отношение $\equiv_L$ имеет конечный индекс и при этом L есть объединение некоторых классов эквивалентности по $\equiv_L$. Можно ли построить конечный автомат, допускающий L? Другими словами, является ли L A-языком?

Эта проблема немедленно сводится к другой, решенной в п. 14.3.3. Действительно, конгруэнция по допустимым контекстам, будучи совместимой с конкатенацией вправо, является эквивалентностью, не менее тонкой, чем эквивалентность по допустимым концам. А поскольку индекс отношения $\equiv_L$ конечен, то индекс отношения $\mathfrak{I}_L$ также оказывается конечным. Стало быть, к $\mathfrak{I}_L$ применима теорема пункта 14.3.3. Мы получаем следующую теорему:

Теорема. *Для того чтобы язык L был автоматным, необходимо и достаточно, чтобы конгруэнция по допустимым контекстам относительно L имела конечный индекс.*

Мы видели, однако, что отношение $\equiv_L$ есть наименее тонкая из всех конгруэнций, относительно которых L является объединением классов. Поэтому для того, чтобы индекс отношения $\equiv_L$ был конечным, достаточно, чтобы был конечен индекс хотя бы одной из таких конгруэнций. Это позволяет нам сформулировать

Следствие. *Для того чтобы язык L был A-языком, необходимо и достаточно, чтобы существовала конгруэнция конечного индекса, относительно которой L был бы объединением классов.*

14.3.6. Формулировка в терминах гомоморфизмов

Произвольная конгруэнция $\mathfrak{R}$ на полугруппе $\mathbf{M}$ определяет факторможество $\mathbf{M}/\mathfrak{R}$ и естественное отображение каждого $A \in \mathbf{M}$ на его класс $\bar{A} \in \mathbf{M}/\mathfrak{R}$.

Пусть $A_1, A_2 \in \bar{A}$ и $B_1, B_2 \in \bar{B}$. Тогда

$$\overline{A_1} = \overline{A_2} \Rightarrow \overline{A_1 B_2} = \overline{A_2 B_2} \qquad \text{(совместимость вправо);}$$

$$\overline{B_1} = \overline{B_2} \Rightarrow \overline{A_1 B_1} = \overline{A_1 B_2} \qquad \text{(совместимость влево).}$$

Отсюда

$$[\overline{A}_1 = \overline{A}_2 \,\&\, \overline{B}_1 = \overline{B}_2] \Rightarrow \overline{A_1B_1} = \overline{A_2B_2}.$$

На множестве классов эквивалентности можно определить операцию, индуцируемую операцией, определенной на $\mathbf{M}$: $\overline{A}\,\overline{B}$ есть класс всех A_iB_i, таких, что $A_i \in \overline{A}$, $B_i \in \overline{B}$. Эта новая операция ассоциативна и превращает $\mathbf{M}/\mathfrak{R}$ в полугруппу.

Естественное отображение $\mathbf{M}$ на $\mathbf{M}/\mathfrak{R}$ (обозначим его через φ) таково, что

$$\varphi(AB) = \varphi(A)\,\varphi(B);$$

следовательно, φ есть гомоморфизм.

Вообще, пусть θ — гомоморфизм, отображающий полугруппу $\mathbf{M}$ на полугруппу $\mathbf{M}'$. По определению гомоморфизма

$$[A \xrightarrow{\theta} A' \,\&\, B \xrightarrow{\theta} B'] \Rightarrow AB \xrightarrow{\theta} A'B'.$$

Пусть отношение $\mathfrak{R}$ определено следующим образом: $A_1 \mathfrak{R} A_2$ тогда и только тогда, когда A_1 и A_2 имеют при отображении θ один и тот же образ. Очевидно, что $\mathfrak{R}$ — эквивалентность; поскольку θ — гомоморфизм, $\mathfrak{R}$ совместимо влево и вправо с операцией полугруппы $\mathbf{M}$ и, следовательно, является конгруэнцией.

Таким образом, между $\mathbf{M}/\mathfrak{R}$ и $\mathbf{M}'$ существует изоморфизм; классу элементов, имеющих в качестве образа A', отвечает при этом изоморфизме само A'.

Прообразы. Пусть $\mathbf{P}'$ — подмножество множества $\mathbf{M}'$. Обозначим через $\theta^{-1}(\mathbf{P}')$ множество тех элементов $\mathbf{M}$, образы которых принадлежат $\mathbf{P}'$. В частности, $\theta^{-1}(A')$ при $A' \in \mathbf{M}'$ есть класс элементов, образом которых является A'. Обозначив этот класс через A, получим

$$\theta^{-1}(\theta(A)) = A.$$

Рассмотрим теперь подмножество $\mathbf{P}$ множества $\mathbf{M}$ и попытаемся найти необходимое и достаточное условие для того, чтобы $\theta^{-1}(\theta(\mathbf{P})) = \mathbf{P}$.

Обозначим через $\mathfrak{R}$ конгруэнцию, определяемую по гомоморфизму θ так же, как и выше. Тогда имеем достаточное условие: $\mathbf{P}$ должно быть объединением классов эквивалентности по $\mathfrak{R}$. Это условие является и необходимым, так как для всякого $A' \in \theta(\mathbf{P})$ множество $\theta^{-1}(\theta(\mathbf{P}))$ содержит все прообразы A'. Тем самым имеет место

Теорема. *Если θ — гомоморфное отображение полугруппы $\mathbf{M}$ на полугруппу $\mathbf{M}'$, то $\theta^{-1}(\theta(\mathbf{P})) = \mathbf{P}$ тогда и только тогда, когда $\mathbf{P}$ является объединением некоторых классов эквивалентности по конгруэнции $\mathfrak{R}$, связанной с θ.*

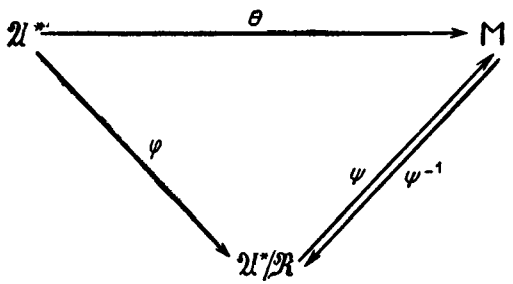
Применение к А-языкам. Для того чтобы язык L в алфавите $\mathfrak{A}$ был автоматным, необходимо и достаточно, чтобы имело место следующее утверждение:

1) На свободной полугруппе $\mathfrak{A}^*$ существует конгруэнция $\mathfrak{R}$, которая имеет конечный индекс и относительно которой L является объединением классов.

Если (1) верно, то факторполугруппа $\mathfrak{A}^*/\mathfrak{R}$, являющаяся образом полугруппы $\mathfrak{A}^*$ при естественном гомоморфизме φ, связанном с конгруэнцией $\mathfrak{R}$, конечна и, кроме того,

$$\varphi^{-1}(\varphi(L)) = L.$$

Вообще, пусть θ — гомоморфизм свободной полугруппы $\mathfrak{A}^*$ на конечную полугруппу $\mathbf{M}$. Тогда конгруэнция $\mathfrak{R}$, ассоциированная с θ, определяет факторполугруппу $\mathfrak{A}^*/\mathfrak{R}$, изоморфную $\mathbf{M}$:



Так как ψ — изоморфизм, из равенства $\theta^{-1}(\theta(L)) = L$ следует, что $\varphi^{-1}(\varphi(L)) = \varphi$, и наоборот. Поэтому утверждение (1) равносильно утверждению (2).

2) Существуют конечная полугруппа $\mathbf{M}$ и гомоморфизм θ свободной полугруппы $\mathfrak{A}^*$ на $\mathbf{M}$, такой, что

$$\theta^{-1}(\theta(L)) = L.$$

Все сказанное выше можно сформулировать в виде следующей теоремы:

Теорема. *Для того чтобы язык $L \subset \mathfrak{A}^*$ был автоматным, необходимо и достаточно, чтобы существовал гомоморфизм θ свободной полугруппы $\mathfrak{A}^*$ на конечную полугруппу $\mathbf{M}$, такой, что*

$$\theta^{-1}(\theta(L)) = L.$$

14.3.7. А-языки и графы

Пусть X — алфавит, и пусть $A, B \subset X$, $V \subset X^2$.

Множество $AX^* \cap X^*B \cap (X^* \setminus X^* VX^*)$ является стандартным А-языком.

Напомним, что это множество состоит из тех цепочек множества X^*, которые

— либо являются однобуквенными и принадлежат $A \cap B$;

— либо начинаются буквой из A, оканчиваются буквой из B и не содержат диграмм, имеющихся в V.

Напомним формальное определение графа.

Графом называется пара $\langle G, R\rangle$, где

1) G — множество элементов, называемых *вершинами*;

2) R — подмножество декартова произведения $G \times G$; элементы R называются *дугами*.

Пусть H и K — подмножества множества G. Назовем *путем из H в K* любую последовательность вершин, которая:

а) либо состоит из одной вершины, принадлежащей $H \cap K$;

б) либо имеет вид $x_1, \ldots, x_i, \ldots, x_n$, где $x_1 \in H$, $x_n \in K$ и

$$(\forall i \in \{1, \ldots, n-1\})[(x_i, x_{i+1}) \in R].$$

Нетрудно видеть, что если отождествить G с X, то множеств путей из H в K совпадет со стандартным А-языком

$$HX^* \cap X^*K \setminus X^*VX^*, \text{ где } V = X^2 \setminus R.$$

§ 14.4. ПРЕОБРАЗОВАНИЯ

14.4.1. Односторонние преобразования

Рассмотрим с некоторой новой точки зрения понятие одностороннего преобразователя (с которым мы встречались ранее в § 10.6) и попытаемся обобщить его; будем говорить при этом не о «преобразователях», а о «преобразованиях».

Пусть $X = \{x_i \mid 1 \leqslant i \leqslant m\}$ — входной алфавит, $Y = \{y_j \mid 1 \leqslant j \leqslant n\}$ — выходной алфавит и $\Sigma = \{S_k \mid 1 \leqslant k \leqslant p\}$ — множество состояний. Пусть, далее, отображение (tr): $\Sigma \times X \xrightarrow{(\mathrm{tr})} \Sigma$ задает функцию переходов: паре (S, x) ставится в соответствие некоторое состояние, обозначаемое Sx. Наконец, пусть имеется отображение η, сопоставляющее паре (S, x) цепочку $\eta(S, x) \in Y$:

$$\Sigma \times X \xrightarrow{\eta} Y.$$

Под *левым односторонним преобразованием, исходящим из состояния S_k*, мы будем понимать отображение X^* в Y^*, которое строится следующим образом. Пусть $f \in X^*$ — данная цепочка:

$$f = x_{i_1} \ldots x_{i_\lambda}.$$

Первой букве x_{i_1} этой цепочки мы поставим в соответствие букву $\eta(S_{k_1}, x_{i_1})$ и состояние $S_{k_2} = S_{k_1}x_{i_1}$.

Второй букве x_{i_2} поставим в соответствие букву $\eta(S_{k_2}, x_{i_2})$ и состояние $S_{k_3} = S_{k_2}x_{i_2}$ и т. д.

Ясно, что таким образом всякой цепочке $f \in X^*$ сопоставляется некоторая цепочка из Y^*.

Для упрощения записи мы будем обозначать начальное состояние через S (без индекса), а образ цепочки f — через $\eta(S, f)$. Выражение $\eta(S, f)$ есть по определению образ цепочки f при преобразовании $[X, Y, \Sigma, (\mathrm{tr}), \eta]$, исходящем из состояния S.

Пусть $\Gamma \subset X^*$. Будем обозначать через $\eta(S, \Gamma)$ множество $\{\eta(S, f) \mid f \in \Gamma\}$ (образ языка Γ).

Пусть, наконец, $\Delta \subset Y^*$ — язык в выходном алфавите; возможно, существуют такие цепочки из X^*, что их образы принадлежат Δ.

Положим

$$\eta^{-1}(S, \Delta) = \{f \in X^* \mid \eta(S, f) \in \Delta\}.$$

NB: не требуется, чтобы все цепочки Δ были образами цепочек из X^*!

14.4.2. Образ языка X^*

Рассмотрим сначала язык $\eta(S, X^*)$ — образ всего языка X^*.

Всякой цепочке $f \in X^*$ соответствует ее образ $\eta(S, f)$ и состояние S_f, в котором заканчивается ее преобразование.

Пусть $g \in Y^*$. Если $g \in \eta(S, X^*)$, то существуют цепочки $f_1, \ldots, f_i, \ldots \in X^*$, для каждой из которых g является образом. Сопоставим цепочке g множество σ_g, состоящее из всех тех состояний $S_{f_1}, \ldots, S_{f_i}, \ldots$, в которых автомат может писать последнюю букву цепочки g. Если $g \notin \eta(S, X^*)$, положим $\sigma_g = \emptyset$. Будем писать $g \sim g'$, если $\sigma_g = \sigma_{g'}$.

1°. Отношение $\sim$ есть отношение эквивалентности.

2°. $\eta(S, X^*)$ представляет собой объединение классов по $\sim$, таких, что $\sigma_g \neq \emptyset$.

3°. Поскольку индекс отношения $\sim$ не больше мощности множества всех подмножеств Σ, он конечен.

4°. Отношение $\sim$ совместимо вправо с конкатенацией.

В самом деле, предположим, что $g \sim g'$, и сравним gh с $g'h$. Если g может быть получено из некоторого f так, что преобразование окончится в состоянии S_α, и если h может быть получено из некоторого u посредством преобразования, начинающегося в состоянии S_α, то существует такое f', что из $f'u$ можно получить $g'h$, причем преобразование окончится в том же состоянии, что и при получении gh из fu.

В силу утверждений 1°, 2° и 4° отношение эквивалентности по допустимым концам для $\eta(S, X^*)$ является не более тонким, чем отношение $\sim$ (по теореме п. 13.2.4). Отсюда и из 3° в силу теоремы п. 14.3.3 следует, что множество $\eta(S, X^*)$ является А-языком.

Итак, мы видим, что $\eta(S, X^*)$ обладает некоторой структурой. Однако подмножество множества $\eta(S, X^*)$ может и не иметь точного прообраза в X^*, если оно не обладает естественной структурой.

14.4.3. Образ произвольного А-языка

Если язык X^* — автоматный, то его образ также является А-языком.

Пусть $\Gamma \subset X^*$ есть А-язык. Мы знаем, что отношение эквивалентности по допустимым концам относительно Γ индуцирует разбиение множества X^* на конечное число классов, причем язык Γ оказывается объединением некоторых из них.

Пусть $f \in \Gamma$. Цепочка f принадлежит некоторому классу γ_f; ее преобразование оканчивается в состоянии S_f. Пусть, далее, $g \in Y^*$. Если $g \in \eta(S, X^*)$, то множество $\{f_1, \ldots, f_t\}$ ее прообразов не пусто; сопоставим в этом случае цепочке g множество пар $\{(\gamma_{f_i}, S_{f_i}) \mid i = 1, \ldots, t\}$. Если $g \notin \eta(S, X^*)$, поставим в соответствие цепочке g пустое множество.

Если множество пар, сопоставленное указанным образом цепочке g, совпадает с множеством пар, сопоставленным таким же образом цепочке g', мы будем писать $g \sim_\Gamma g'$.

1°. Отношение $\sim_\Gamma$ является эквивалентностью.

2°. Множество $\eta(S, \Gamma)$ представляет собой объединение классов, отвечающих таким множествам $\{(\gamma_{f_1}, S_{f_1}), \ldots, (\gamma_{f_t}, S_t)\}$, для которых хотя бы один класс γ_i содержится в Γ.

3°. Индекс отношения $\sim_\Gamma$ конечен.

4°. Отношение $\sim_\Gamma$ совместимо с конкатенацией вправо.

Для доказательства утверждения 4° достаточно вспомнить то, что говорилось при доказательстве аналогичного утверждения в предыдущем пункте, и добавить, что класс эквивалентности (по допустимым концам) цепочки fu совпадает с классом цепочки $f'u$, если только это верно для f и f'.

Из 1°—4° следует, что $\eta(S, \Gamma)$ есть А-язык. Нетрудно видеть, что классы эквивалентности, определяемые этим языком, содержатся в естественных классах эквивалентности, определяемых языком $\eta(S, X^*)$ (см. предыдущий пункт). Таким образом, если некоторый А-язык $\Delta \subset Y^*$ определяет классы, не содержащиеся в «естественных» классах, то Δ не может быть образом никакого А-языка $\Gamma \subset X^*$.

14.4.4. Прообраз А-языка

Если $\Delta \subset Y^*$ есть А-язык, то множество цепочек, образы которых принадлежат Δ, также является А-языком.

Это доказывается путем рассмотрения следующей эквивалентности: $f \sim_\Delta f'$ тогда и только тогда, когда класс эквивалентности $\eta(S, f)$ по допустимым концам относительно Δ совпадает с классом эквивалентности цепочки f'.

14.4.5. Еще о языке $\eta(S, X^*)$

Мы видели, что $\eta(S, X^*)$ есть А-язык. Можно поставить следующий вопрос: если дан алфавит Y и А-язык $L \subset Y^*$, то всегда ли

можно построить такое преобразование, чтобы $\eta(S, X^*)$ совпало с L?

Ответ на этот вопрос оказывается отрицательным. Пусть, например, $Y = \{a, b\}$; нетрудно убедиться, что не существует преобразования, которое отображало бы X^* на a^*b (чтобы построить такое преобразование, пришлось бы отказаться от детерминированности).

14.4.6. Произведение односторонних преобразований

Определим *правое одностороннее перобразование* так же, как мы определяли левое одностороннее преобразование; для этого достаточно изменить направление чтения. Отображение, задающее функцию переходов для правого преобразования, мы будем записывать следующим образом:

$$(\mathrm{tr}'):\ X \times \Sigma' \xrightarrow{(\mathrm{tr}')} \Sigma'.$$

Отображение (tr′) ставит в соответствие паре (x, S') состояние, которое мы будем обозначать через xS'.

Теперь можно рассматривать композицию односторонних преобразований, как левых, так и правых: первое отображает X^* в Y^*, второе — Y^* в Z^* и т. д. Результат композиции преобразований называется их *произведением*.

Имеет место следующий результат: если X и Z — произвольные алфавиты и $L \subset Z^*$ есть А-язык, то существуют такое левое преобразование и такое правое преобразование, что их произведение отображает X^* на L.

Данное утверждение доказывается сначала для стандартного А-языка $IZ^* \cap Z^*J \setminus Z^*VZ^*$ (напомним, что I — множество начальных букв, J — множество заключительных букв, V — множество запрещенных диграмм). Затем следует перейти к общему случаю с помощью гомоморфизма, описанного в п. 14.2.3.

14.4.7. Двусторонние преобразования

Чтобы определить *двустороннее преобразование* множества X^* в множество Y^*, нужно задать две функции переходов для двух множеств состояний:

$$(\mathrm{tr}):\ \Sigma \times X \to \Sigma,$$
$$(S, x) \to Sx;$$
$$(\mathrm{tr}'):\ X \times \Sigma' \to \Sigma',$$
$$(x, S') \to xS'.$$

Кроме того, должно быть задано следующее отображение:

$$\eta:\ (\Sigma, X, \Sigma') \to Y,$$
$$(S, x, S') \to \eta(S, x, S').$$

Цепочка $y_{i_1} \dots y_{i_r}$ будет по определению образом цепочки $x_{i_1} \dots x_{i_r}$ при двустороннем преобразовании, исходящем из состояний S и S' ($x_{i_1}, \dots, x_{i_r} \in X$, $y_{i_1}, \dots, y_{i_r} \in Y$, $S \in \Sigma$, $S' \in \Sigma'$), если найдутся такие последовательности состояний $S_0, \dots, S_r \in \Sigma$, $S'_0, \dots, S'_r \in \Sigma'$, что

а) $S_0 = S$, $S_1 = S_0 x_{i_1}$, $\dots$, $S_r = S_{r-1} x_{i_r}$;

б) $S'_r = S'$, $S'_{r-1} = x_{i_r} S'_r$, $\dots$, $S'_0 = x_{i_1} S'_1$;

в) $y_{i_1} = \eta(S_0, x_{i_1}, S'_1)$, $y_{i_2} = \eta(S_1, x_{i_2}, S'_2)$, $\dots$, $y_{i_r} = \eta(S_{r-1}, x_{i_r}, S'_r)$.

14.4.8. Двусторонние преобразования и А-языки

Результаты, полученные для случая одностороннего преобразования, можно обобщить и на случай двустороннего преобразования: если Γ есть А-язык, то и $\eta(S, \Gamma, S')$ есть А-язык; если $\Delta \subset Y^*$ есть А-язык, то и $\eta^{-1}(S, \Delta, S')$ — также А-язык.

Для доказательства этих утверждений полезно представить преобразование с помощью графа, вершинами которого служат пары (S, S'), а из (S_i, S'_k) в (S_α, S'_γ) идет дуга, помеченная парой (x_j, y_l), тогда и только тогда, когда $S_\alpha = S_i x_j$, $S'_i = x_j S'_\gamma$, $y_l = \eta(S_i, x_j, S'_\gamma)$.

14.4.9. Упражнения

1. *Преобразование, переводящее X^* в заданный стандартный А-язык.*

Построим сначала левое одностороннее преобразование, отображающее X^* в $(X \cup \{\omega\})^*$, где ω — некоторый специальный символ. Каждому $x_i \in X$ будет отвечать состояние $[x_i]$; кроме того, будут еще два состояния S_1 и S_2.

Правила перехода из одного состояния в другое таковы:

$$\begin{aligned} S_1 x &= [x] \quad \text{для} \quad x \in I \\ S_1 x &= S_2 \quad \text{для} \quad x \notin I \\ [x_1]\, x_2 &= [x_2] \quad \text{для} \quad x_1 x_2 \notin V \\ [x_1]\, x_2 &= S_2 \quad \text{для} \quad x_1 x_2 \in V. \end{aligned}$$

Отображение η определяется следующим образом:

$$\begin{aligned} \eta(S_1, x) &= e \quad \text{для} \quad x \in I \\ \eta(S_1, x) &= x \quad \text{для} \quad x \notin I \\ \eta(S_2, x) &= e \\ \eta([x_i], x_j) &= \omega \quad \text{для} \quad x_i x_j \in V \\ \eta([x_i], x_j) &= x_j \quad \text{для} \quad x_i x_j \notin V. \end{aligned}$$

Полученный таким образом язык следует перевести в заданный стандартный язык с помощью правого одностороннего преобразования (при построении которого должно быть учтено J).

2. Пусть η — левое одностороннее преобразование множества X^* в Y^* и μ — правое одностороннее преобразование Y^* в Z^*.

Всегда ли существуют правое одностороннее преобразование μ' и левое одностороннее преобразование η', такие, что

$$\eta\mu = \mu'\eta'?$$

Глава XV

ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О КС-ЯЗЫКАХ

§ 15.1. ЯЗЫКИ ДИКА

15.1.1. Свободная группа

Пусть $\mathfrak{A} = \{a, b, \ldots\}$ и $\mathfrak{A}' = \{a', b', \ldots\}$ — два конечных непересекающихся алфавита с одинаковым количеством букв. Пусть между этими алфавитами установлено взаимно однозначное соответствие: $a — a'$, $b — b'$ и т. д. Пусть, кроме того, $\mathfrak{B} = \mathfrak{A} \cup \mathfrak{A}'$.

Обозначив через E пустое слово, зададим следующую систему $\mathfrak{R}$ соотношений Туэ:

$$\mathfrak{R}: \; aa' \sim a'a \sim E, \; bb' \sim b'b \sim E, \ldots .$$

Эти соотношения определяют отношение эквивалентности; мы знаем, что $\mathfrak{B}^*/\mathfrak{R}$ — полугруппа с единицей.

Эта полугруппа является группой. В самом деле, для всякого слова A можно найти слово $\bar{A}$, такое, что

$$A\bar{A} \sim \bar{A}A \sim E.$$

Для этого достаточно

. взять слово $\tilde{A}$ — обращение слова A;

.. «инвертировать» штриховку в $\tilde{A}$ (т. е. снять штрихи у тех букв, где они были, и поставить у тех, где их не было).

Пример. Для $A = aba'b'abb$ имеем

$$\tilde{A} = bbab'a'ba,$$

$$\bar{A} = b'b'a'bab'a'.$$

Ясно, что элементом $\mathfrak{B}^*/\mathfrak{R}$, обратным классу элемента A, будет класс элемента $\bar{A}$.

Покажем теперь, что группа $\mathfrak{B}^*/\mathfrak{R}$ изоморфна свободной группе, порождаемой множеством $\mathfrak{A}$, и определим канонический гомоморфизм, отображающий $\mathfrak{B}^*$ на эту группу.

15.1.2. Представление произвольного класса из $\mathfrak{B}^*/\mathfrak{R}$

Будем называть слово *несократимым*, если в нем нет ни одного вхождения буквы x, смежного с вхождением буквы x'.

Пример. Слово $aba'b'ab$ несократимо; слово $abb'ab$ сократимо Пустое слово является несократимым.

Предложение. *Всякий класс определенной выше эквивалентности содержит одно и только одно несократимое слово.*

Существование несократимого слова в каждом классе очевидно. Докажем его единственность.

Для этого покажем сначала, что справедлива

Лемма 1. *Единственное несократимое слово, эквивалентное пустому слову, есть само пустое слово.*

Будем говорить, что слова x и y смежные, если от одного из них к другому можно перейти однократным применением одного соотношения.

Выражение «слово A эквивалентно пустому слову E» означает, что имеется последовательность слов $x_0, x_1, \ldots, x_n$, такая, что $x_0 = A$, $x_n = E$ и каждое x_i смежно с x_{i-1}. Поскольку по этой последовательности можно двигаться и в обратном направлении, то, образуя все слова, смежные с E, затем все слова, смежные с этими смежными, и т. д., мы обязательно на каком-то шаге получим A. Словами, смежными с E, являются aa', $a'a$, bb' и т. д. Чтобы получить слова, смежные с этими последними, можно либо сокращать их, но тогда мы будем вновь получать E, либо вставлять в любое место одно из слов aa', $a'a$, bb', $b'b$, ...

Вообще, строя слова, смежные с данными словами, следует «наращивать», а не сокращать, потому что любое сокращение возвращает нас к слову, уже получившемуся на одном из предыдущих шагов [1]).

---

[1]) Точнее, всякое слово, эквивалентное E, может быть получено из E одними вставками (без сокращений). Это утверждение, из которого немедленно вытекает лемма 1, может быть доказано следующим образом Пусть $A_0, \ldots, A_n$ — последовательность слов, в которой каждое следующее слово смежно с предыдущим, и при этом $A_0 = E$ Пусть при определяемом этой последовательностью способе перехода от E к A_n первое сокращение происходит на k-м шаге, т е. при получении A_k из A_{k-1}. Тогда либо $A_{k-1} = Xaa'Y$, $A_k = XY$, либо то же с перестановкой a и a'; можно ограничиться разбором первого случая Если данные («главные») вхождения a и a' были вставлены на одном шаге, этот шаг, так же как и k-й, можно было бы не делать, и мы получаем способ перехода от E к A_n с меньшим числом сокращений. Пусть эти вхождения были вставлены на разных шагах. Тогда возможны три случая: 1) $X = X_1a'X_2$, $Y = Y_2aY_1$, причем выделенное вхождение a' в X было вставлено вместе с «главным» вхождением a на i-м шаге, а выделенное вхождение a в Y — вместе с «главным» вхождением a' на j-м. Пусть для определенности $i < j$. Тогда можно не делать i-го и k-го шагов, а все те вставки, в результате которых получается Y_2 (все они делались после j-го шага), производить не правее вставленного на j-м шаге a', а левее В результате снова получаем способ перехода от E к A_n с меньшим числом вставок. 2) $X = X_1aX_2a'X_3$, причем выделенное вхождение a было вставлено вместе с «главным» вхождением a' на i-м шаге, а выделенное вхождение a' — вместе с «главным» вхождением a на j-м, $i < j$. В этом случае можно опять-таки не делать i-го и k-го шагов, а те вставки, которые дают X_3 (они делались после j-го шага), делать не левее вставленного на j-м шаге a', а правее. 3) Случай, когда оба вхождения a' и a, парных «главным» вхождениям, находятся в Y, аналогичен предыдущему. — *Прим. ред*

Пример. Начиная со слова $aa'bb'$ и вставляя aa' между aa' и bb', мы получим $aa'aa'bb'$. Если мы будем сокращать, как показано подчеркиванием: $\underline{aa'}aa'bb'$, то все равно вернемся к $aa'bb'$.

Из сказанного следует, что в классе слова E не существует непустых несократимых слов.

Теперь нам понадобится еще

Лемма 2. *Если бы существовали различные несократимые слова, эквивалентные друг другу, то существовали бы непустые несократимые слова, эквивалентные пустому слову.*

Пусть A и B — несократимые и эквивалентные друг другу слова. Будем считать, что они начинаются с разных букв (в противном случае этого было бы легко добиться посредством сокращений)[1]):

$$A = xA_1,\ B = yB_1,\ x \neq y.$$

Ясно, что при данных условиях слово $\bar{A}B = \bar{A}_1\bar{x}_1yB_1$ несократимо.

Но из $A \sim B$ следует $\bar{A}B \sim \bar{A}A \sim E$, что и требовалось доказать.

Наше предложение вытекает теперь непосредственно из лемм 1 и 2.

15.1.3. Канонический гомоморфизм

Будем считать известным определение свободной группы с системой свободных образующих $\mathfrak{A}$ (см. упражнение 2 из п. 15.1.10).

Пусть дан канонический гомоморфизм γ, отображающий $\mathfrak{B}^*$ на эту свободную группу:

$$E \to 1,$$
$$a \to a,$$
$$a' \to a^{-1};$$

тогда из $\gamma(f) = \gamma(f')$ следует, что f и f' сокращаются до одного и того же несократимого слова.

Множество $\gamma^{-1}(1)$, т. е. ядро гомоморфизма γ, представляет собой множество слов, сократимых до E. Это множество называют *языком Дика*, определенным над $\mathfrak{B} = \mathfrak{A} \cup \mathfrak{A}'$; мы будем обозначать его через D^*.

15.1.4. Ограниченные языки Дика

Если выполнять только сокращения пар aa', bb' и т. п., но не пар $a'a, b'b, \ldots$, то получается *ограниченный язык Дика*[2]).

---

[1]) Точнее, из $aX \sim aY$ следует $X \sim Y$. Действительно, $aX \sim aY$ влечет за собой $a'aX \sim a'aY$, но $a'aX \sim X$, $a'aY \sim Y$. — *Прим. ред.*

[2]) Точнее, ограниченный язык Дика над алфавитом $\mathfrak{B} = \mathfrak{A} \cup \mathfrak{A}'$ есть по определению множество тех слов (цепочек) в алфавите $\mathfrak{B}$, которые переводятся в E последовательным вычеркиванием пар $aa', bb', \ldots$. — *Прим. ред.*

Как уже отмечалось (см. п. 1.3.2), ограниченные языки Дика хорошо известны в математике: достаточно интерпретировать буквы без штрихов как левые скобки, а буквы со штрихами — как соответствующие их правые скобки.

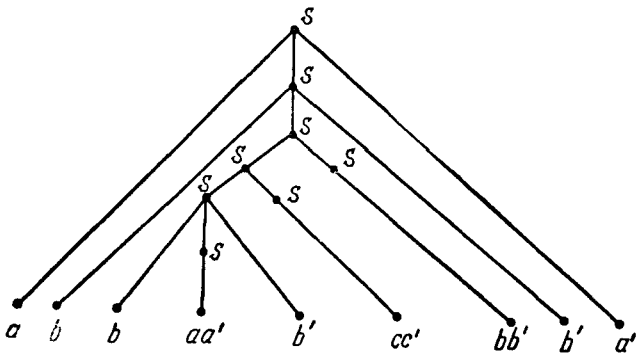
15.1.5. Языки Дика являются КС-языками

Интерпретация букв как скобок и ее «обобщение» на случай произвольных языков Дика делают очевидным тот факт, что языки Дика являются КС-языками.

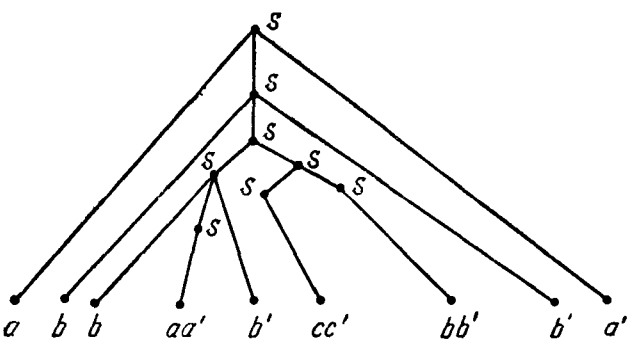
Ограниченный язык Дика может быть порожден, например, следующей грамматикой:

$$S \to SS$$
$$S \to aSa',\ bSb',\ \ldots$$
$$S \to aa',\ bb',\ \ldots$$

Например, для слова (цепочки) $abbaa'b'cc'bb'b'a'$ имеется вывод



Однако эта цепочка имеет и другой вывод:



Таким образом, построенная нами грамматика является неоднозначной. Рассмотрим теперь языки Дика более подробно с тем,

чтобы предложить для них однозначные грамматики, более согласованные с их структурой.

15.1.6. Разложение цепочек на Д-простые цепочки

Будем говорить, что цепочка $d \in D^*$ является *Д-простой* (или *простой по Дику*), если никакое непустое начало цепочки d, отличное от самой d, не принадлежит D^* [1]). Заменяя слово «начало» словом «конец», получим эквивалентное определение [2]).

Пример. Цепочка $a'baa'bb'b'a$ является Д-простой; цепочка $abb'a'bb'$ не является Д-простой.

Теорема. *Всякая цепочка, принадлежащая языку Дика, единственным образом представима как произведение (конкатенация) Д-простых цепочек.*

Для Д-простой цепочки теорема очевидна. Пусть d не является Д-простой цепочкой; предположим, что она может быть разложена двумя разными способами:

$$d = f_1 f_2 \ldots f_\lambda = g_1 g_2 \ldots g_\mu.$$

Если $|f_1| = |g_1|$, то $f_1 = g_1$; сокращая, получим $f_2 \ldots f_\lambda = g_2 \ldots g_\mu$, после чего рассуждение можно повторить.

Если $|f_1| < |g_1|$, то g_1 не является Д-простой цепочкой.

15.1.7. Структура Д-простых цепочек

Пусть цепочка $g = xfy$, где x и y — буквы, является Д-простой. В ходе сокращения цепочки g буква x сокращается с y и не может сокращаться ни с какой другой буквой; действительно, в противном случае цепочка g имела бы непустое начало, принадлежащее языку Дика и отличное от самой g. Тем самым y «спарена» с x, и мы будем писать $y = \bar{x}$.

Таким образом, цепочка f принадлежит языку Дика, и, следовательно, ее можно представить как произведение Д-простых цепочек:

$$f = f_1 f_2 \ldots f_m,$$

где f_1 имеет вид $x_1 f_1' \bar{x}_1$, причем $x_1 \neq \bar{x}$, поскольку в противном случае цепочка g не была бы Д-простой. Аналогично получаем: $f_2 = f_2' \bar{x}_2$, где $x_2 \neq \bar{x}$ [3]), $f_3 = f_3' \bar{x}_3$, где $x_3 \neq \bar{x}$, и т. д.

Обозначим через D_y множество всех Д-простых цепочек, начинающихся буквой y (и оканчивающихся буквой $\bar{y}$).

---

[1]) *Началом*, соответственно *концом*, цепочки X называется всякая цепочка Y, такая, что X представима в виде YZ, соответственно ZY. — *Прим. ред.*

[2]) Действительно, если цепочка $X \in D^*$ представима в виде YZ, где $Y \in D^*$, то $Z \in D^*$ (иначе, если $X = YZ$ и Y сократимы до E, то и Z сократимо до E), и обратно. — *Прим. ред.*

[3]) Действительно, в противном случае цепочка xf_1x_2, являющаяся непустым началом g, совпадала бы с $xf_1\bar{x}$ и тем самым принадлежала бы языку Дика. — *Прим ред.*

Мы можем теперь сказать, что всякая Д-простая цепочка имеет единственное представление вида

$$g = xf_1 \ldots f_m\bar{x},$$

где $f_i \in D_{x_i}$, $x_i \neq \bar{x}$, $i = 1, \ldots, m$.

Пример. Цепочка $ab'cc'baa'c'a'aca'$ записывается следующим образом:

$$a\ b'cc'b\ aa'\ c'a'ac\ a'.$$

Обозначим через D множество всех Д-простых цепочек. В силу теоремы о единственности разложения на Д-простые цепочки язык D^* есть свободная полугруппа над D (этим объясняются используемые обозначения).

15.1.8. Однозначная грамматика, порождающая язык Дика

Построим для языка Дика однозначную КС-грамматику; она будет записана в виде системы уравнений.

Для простоты ограничимся алфавитом $\{a, a', b, b'\}$, что никак не отразится на общности наших рассуждений.

Введем нетерминальные символы D_a, $D_{a'}$, D_b, $D_{b'}$, имеющие то же значение, что и в предыдущем пункте.

D_a содержит, помимо цепочки aa', все цепочки, имеющие вид

$$af_1 \ldots f_ma',$$

где $f_i \in D_{x_i}$, $x_i \neq \bar{a}$, и только такие цепочки.

Чтобы выразить этот факт, воспользуемся уравнением

$$D_a = aAa' + aa', \tag{1a}$$

где A — еще один нетерминальный символ, соответствующий некоторому языку. Этот язык содержит, в частности, $D_{a'}$, D_b, $D_{b'}$; точнее, он состоит из конкатенаций любых цепочек, принадлежащих $D_{a'} \cup D_b \cup D_{b'}$. Следовательно, мы можем написать уравнение

$$A = (D_{a'} + D_b + D_{b'})(A + E), \tag{2a}$$

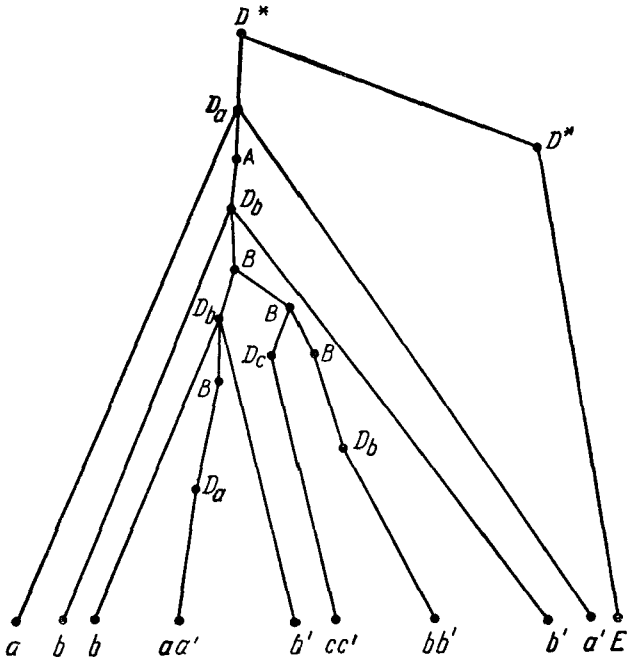
где E — пустая цепочка

Что касается языка Дика D, то он соответствует уравнению

$$D^* = (D_a + D_{a'} + D_b + D_{b'})\, D^* + E. \tag{3}$$

Уравнения типов (1) и (2) вместе с уравнением (3) задают некоторую грамматику языка Дика. Эта грамматика однозначна, поскольку она порождает слева направо Д-простые сомножители цепочки, принадлежащей D^*.

Пример. Возьмем цепочку $abbaa'b'cc'bb'b'a'$, уже фигурировавшую в п. 15.1.5. В построенной нами грамматике эта цепочка имеет только один вывод [1]):



15.1.9. Однозначная грамматика, порождающая ограниченный язык Дика

При построении грамматики для ограниченного языка Дика достаточно рассмотреть множества D_a, D_b и т. д., исключив $D_{a'}$, $D_{b'}$, ... Теперь A, B, ... можно не различать, поскольку цепочки, начинающиеся штрихованными буквами, не рассматриваются. В результате мы имеем

$$D_a = aUa' + aa',$$

$$D_b = bUb' + bb',$$

$$U = (D_a + D_b)(U + E),$$

$$D_r^* = (D_a + D_b) D_r^* + E.$$

Аксиомой является D_r^* («ограниченный язык Дика»).

---

[1]) Для простоты в дереве вывода не показаны пути, оканчивающиеся в E (кроме одного); эти пути не изменяют терминальную цепочку. — *Прим. ред.*

Пример. Сюда подходит предыдущий пример, в котором сознательно была взята цепочка из D_r^*; надо только заменить A и B на U.

15.1.10. Упражнения

1. Пусть имеется алфавит X (содержащий как «нештрихованные», так и «штрихованные» буквы). Обозначим через $\bar{f}$ цепочку, которая получается из f обращением этой последней с последующим инвертированием штриховки. Положим

$$U = \{f\bar{f} \mid f \neq E\},$$

$$V = \{v \in U \mid v \notin UXX^*\}.$$

Иначе говоря, V состоит из цепочек языка U, никакие собственные начала которых не принадлежат U.

Доказать, что всякая цепочка $f\bar{f} \in U$ единственным образом представляется как конкатенация цепочек из V.

2. *Определение свободной группы.* Пусть $\mathfrak{A} = \{a, a', b, b', \ldots\}$ — алфавит, состоящий из $2n$ «спаренных» букв (как все алфавиты языков Дика; ср. п. 1.3.2). Пусть, далее, имеются соотношения Туэ: $aa' = a'a = E$ и т. д.

Для того чтобы слово в алфавите $\mathfrak{A}$ было несократимо, необходимо и достаточно, чтобы ни одно вхождение буквы x_i не было смежным с вхождением «парной» буквы x_i'.

Рассмотрим множество несократимых слов, включив в него пустое слово E. Снабдим это множество операцией, которую мы будем обозначать точкой и называть умножением, следующим образом.

Чтобы получить произведение $X \cdot Y$, нужно

1°. образовать конкатенацию XY;

2°. рассмотреть пару вхождений букв на стыке X и Y;

— если эти буквы не «спаренные», то $X \cdot Y = XY$;

— если они «спаренные», нужно вычеркнуть данную пару вхождений;

— затем нужно проделать то же самое с буквами «на стыке» слов, оставшихся от X и Y, и т. д. Кроме того, полагаем $X \cdot E = E \cdot X = X$.

Примеры.

$X = aba'cb$, $Y = c'$, $X \cdot Y = aba'cbc'$;

$X = aba'$, $Y = ab'$, $X \cdot Y = a$;

$X = bac'$, $Y = ca'b'$, $X \cdot Y = E$.

Доказать, что множество несократимых слов, снабженное таким умножением, образует группу.

Поскольку не очевидна лишь ассоциативность, наметим путь ее доказательства.

Пусть X, Y, Z — несократимые слова; докажем равенство

$$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z).$$

Если хотя бы одно из слов X, Y, Z пусто, то равенство очевидно. Поэтому можно сначала доказать его для случая, когда Y — однобуквенное слово, а затем воспользоваться индукцией по длине Y: допустим, что для $|Y| = m - 1$ равенство доказано; возьмем $Y = Y_1 y$, где $|Y_1| = m - 1$ и $y \in \mathfrak{A}$; имеем $X \cdot (Y \cdot Z) = X \cdot [(Y_1 \cdot y) \cdot Z] = X \cdot [Y_1 \cdot (y \cdot Z)]$ и т. д.

§ 15.2. СТАНДАРТНЫЕ КС-ЯЗЫКИ

15.2.1. Определение

Стандартным КС-языком C_S называется язык, представляющий собой пересечение языка Дика D^* и стандартного А-языка Q:

$$C_S = D^* \cap Q.$$

В п. 10.6.3 было показано, что пересечение КС-языка и А-языка является КС-языком; этот факт оправдывает только что сформулированное определение. Там же была указана процедура, позволяющая построить для указанного пересечения КС-грамматику. Эта грамматика будет однозначной, если таковы грамматики исходных языков D^* и Q.

Примеры. Пусть имеется алфавит $\mathfrak{A} = \{a, b, a', b'\}$; Q определяется начальными буквами $\{a, b\}$ и запретом диграмм вида $x'y$ (т. е. переходов от буквы со штрихом к букве без штриха).

Пересечение языка Q с языком Дика в алфавите $\mathfrak{A}$ имеет вид

$$C_S = \{f \bar{f} \mid f \in \{a, b\}^*\},$$

где $\bar{f}$ означает то же, что в упр. 1 из п. 15.1.10.

15.2.2. Особенности цепочек, принадлежащих стандартным КС-языкам

Цепочки стандартных КС-языков обладают особенностями двух типов, существенно различными по своей природе.

Те особенности, которые восходят к стандартным А-языкам, сугубо локальны. Для проверки их выполнения достаточно линейно ограниченного конечного автомата: ему придется хранить в памяти ровно одну букву (прочитанную последней).

Особенности, восходящие к языкам Дика, должны соблюдаться, так сказать, на неограниченном расстоянии. Для их проверки требуется автомат с неограниченной памятью. Правда, эти особенности касаются «неперекрещивающихся» пар вхождений символов. они отражают скобочную структуру языков Дика.

Итак, мы видим, что стандартные КС-языки имеют достаточно простую и интуитивно ясную структуру. Уже это делает их интересными; однако значение этих языков определяется в основном следующей теоремой.

15.2.3. Основная теорема о КС-языках

Всякому КС-языку L можно сопоставить язык Дика D^* (и даже ограниченный язык Дика D_r^*), стандартный А-язык Q и гомоморфизм φ, такие, что

$$L = \varphi(D^* \cap Q) \text{ (соответственно } L = \varphi(D_r^* \cap Q)).$$

Иначе говоря, всякий КС-язык может быть получен посредством гомоморфного отображения некоторого надлежащим образом выбранного стандартного КС-языка.

Пример. Язык $\{f\bar{f}\}$ в алфавите $\{a, b\}$ получается из стандартного КС-языка $\{f\bar{f}\}$, рассматривавшегося в предыдущем примере, посредством гомоморфизма

$$\varphi: \left| \begin{array}{l} a \to a \\ a' \to a \\ b \to b \\ b' \to b \end{array} \right|.$$

Для доказательства этой теоремы нам нужно ввести понятие простой грамматики и доказать два вспомогательных предложения.

Определение. Назовем КС-грамматику *простой*, если все ее нетерминальные правила имеют вид

$$\xi_l = \sum_l d_{l,\,l} + \sum_{j,\,k} a_{l,\,j,\,k} b_{l,\,j,\,k} \xi_j \bar{b}_{l,\,j,\,k} c_{l,\,j,\,k} \xi_k \bar{c}_{l,\,j,\,k} \bar{a}_{l,\,j,\,k};$$

здесь латинские буквы обозначают разные (NB!) терминальные символы, а греческие — нетерминальные символы.

Предложение 1. *Для всякой грамматики G, порождающей язык L, существует простая грамматика G' и гомоморфизм φ, такие, что $L = \varphi(L(G'))$.*

Дадим набросок доказательства. Если в G имеется правило вида

$$\xi_l \to u\xi' v\xi'' w\xi''',$$

где u, v и w — терминальные цепочки, то вводится нетерминальный символ ζ и правила

$$\xi_l \to \zeta w \xi''',$$

$$\zeta \to u\xi' v\xi''.$$

Если в G имеется правило

$$\xi_i \to u\xi' v,$$

то вводится нетерминальный символ η и правила

$$\xi_i \to u\xi'\eta,$$
$$\eta \to v.$$

Таким образом мы добьемся, чтобы все правила, содержащие в правых частях нетерминальные символы, имели вид

$$\xi_i \to u\xi v\xi' w.$$

Для каждой тройки (u, v, w) введем новые символы a, b, c и определим гомоморфизм φ так, чтобы

$$\varphi(a) = u, \qquad \varphi(\bar{a}) = e,$$
$$\varphi(b) = e, \qquad \varphi(\bar{b}) = v,$$
$$\varphi(c) = e, \qquad \varphi(\bar{c}) = w$$

(e — пустая цепочка). Затем мы можем включить в G' правила

$$\xi_i \to ab\xi\bar{b}c\xi'\bar{c}\bar{a};$$

гомоморфизм φ превратит их в правила грамматики G, и мы будем иметь $L = \varphi(L')$.

Теперь мы можем приступить к доказательству основной теоремы.

Пусть G есть КС-грамматика, порождающая язык L; построим сначала простую грамматику G' в соответствии с предложением 1. Грамматике G' мы сопоставим стандартный А-язык R и язык Дика D^*.

. В языке R разрешены диграммы следующего вида:

$$a_{i,j,k}b_{i,j,k}; \qquad b_{i,j,k}d_{j,i}; \qquad b_{i,j,k}a_{j,i,m};$$
$$c_{i,j,k}d_{k,i}; \qquad c_{i,j,k}a_{k,i,m};$$
$$\bar{a}_{j,i,m}\bar{b}_{i,j,k}; \qquad \bar{a}_{i,j,k}\bar{c}_{i,m,i};$$
$$\bar{b}_{i,j,k}c_{i,j,k}; \qquad \bar{c}_{i,j,k}\bar{a}_{i,j,k};$$
$$d_{i\,i'}\bar{b}_{i\,j,m}; \qquad d_{i,i'}\bar{c}_{i,m,i}.$$

Здесь $a_{i,j,k}$, $b_{i,j,k}$, $c_{i,j,k}$ — буквы, а d_{il} следует понимать как двубуквенную цепочку $\delta_{i,l}\bar{\delta}_{l,i}$ (при этом некоторые диграммы превращаются в триграммы, но разрешение таких триграмм легко свести к разрешению диграмм).

.. D^* есть язык Дика в алфавите $X \cup X'$, где

$$X = \{a_{i,j,k},\ b_{i,j,k},\ c_{i,j,k},\ \delta_{i,l}\}.$$

Основная теорема оказывается теперь непосредственным следствием следующего предложения:

Предложение 2. *Пусть имеется простая грамматика G с правилами вида*

$$\xi_i = \sum_l d_{i,l} + \sum_{j,k} a_{i,j,k} b_{i,j,k} \xi_j \bar{b}_{i,j,k} c_{i,j,k} \xi_k \bar{c}_{i,j,k} \bar{a}_{i,j,k};$$

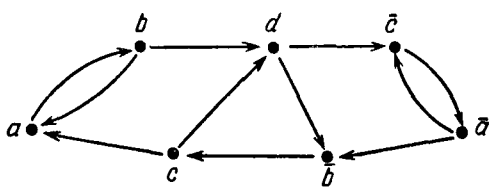
пусть имеются, далее, язык Дика D^* и А-язык R, сопоставленные указанным выше способом грамматике G. Тогда для каждого i язык L_i', состоящий из цепочек, выводимых в G из ξ_i, удовлетворяет следующему уравнению:

$$L_i' = D^* \cap R \cap \left\{ \bigcup_{j,k} a_{i,j,k} X^* \bigcup_l d_{i,l} \right\}.$$

Доказательство удобно провести в два этапа; наметим их. *Первый этап.* Рассмотрим сначала грамматику

$$\xi = d + ab\xi\bar{b}c\xi\bar{c}\bar{a},$$

полученную из G путем отбрасывания индексов и сохранения лишь одного нетерминального символа. Докажем предложение 2 для этого частного случая. А-язык R можно представить посредством следующей диаграммы:



L' состоит из d (т. е. $\delta\bar{\delta}$) и некоторого множества Д-простых цепочек, начинающихся буквой a и удовлетворяющих ограничениям на сочетаемость букв, задающим язык R.

Пусть $L'' = D^* \cap R \cap \{aX^* \cup d\}$. Требуется доказать, что $L' = L''$.

Включение $L' \subset L''$ очевидно. Обратное включение докажем по индукции, воспользовавшись тем, что L'' состоит из d и Д-простых цепочек вида $abA_1\bar{b}cA_2\bar{a}$, где A_1 и A_2 суть Д-простые цепочки такого же вида (быть может, равные d). Доказательство того, что каждая такая цепочка принадлежит L', будет проводиться индукцией по «сложности» цепочки (начиная с d). При этом используется тот факт (п. 15.1.7), что всякая Д-простая цепочка имеет единственное представление вида $af_1 \ldots f_m\bar{a}$, где $f_l \in D_{x_l}$.

Второй этап. Можно убедиться, что предыдущее доказательство сохраняет силу и для случая, когда буквы a, b, c, d имеют индексы.

15.2.4. Следствия из основной теоремы

Мы показали, что класс КС-языков не замкнут относительно операций взятия дополнения и пересечения; было, кроме того, установлено, что не все КС-языки являются детерминированными.

Алгебраическая характеристика КС-языков, доставляемая основной теоремой (п. 15.2.3), объясняет такое положение вещей. Рассмотрим подкласс стандартных КС-языков $D_r \cap Q$, где D_r — класс ограниченных языков Дика, а Q — класс стандартных А-языков на некоторой свободной полугруппе. В этих КС-языках все нерегулярности, связанные с замкнутостью или с допустимостью относительно заданного автомата, пропадают. В случае линейных стандартных КС-языков также ясно, что их пересечение (построение которого сводится к образованию пересечения регулярных языков) является линейным стандартным КС-языком. Кроме того, легко видеть, что эти языки являются детерминированными.

Нерегулярности, связанные с общим классом КС-языков, возникают при гомоморфном отображении φ. Вот один из результатов, относящихся к соответствующим алгоритмически неразрешимым проблемам.

Пусть $\mathfrak{A}^*$ и $\mathfrak{B}^*$ — свободные полугруппы, φ и ψ — гомоморфизмы, отображающие $\mathfrak{A}^*$ в $\mathfrak{B}^*$. Не существует *общей* процедуры, позволяющей установить, существует ли цепочка $f \in \mathfrak{A}^*$, такая, что $\varphi(f) = \psi(f)$. Это просто алгебраическая формулировка результата Поста о неразрешимости проблемы соответствий (п. 6.2.3).

Этот результат можно усилить, распространив его на случай, когда φ и ψ — мономорфизмы, т. е. когда разные f имеют разные образы. Этот факт объясняет неразрешимость проблемы пересечения.

§ 15.3. СОВПАДЕНИЕ КЛАССА КС-ЯЗЫКОВ С КЛАССОМ ЯЗЫКОВ, ДОПУСКАЕМЫХ АВТОМАТАМИ С МАГАЗИННОЙ ПАМЯТЬЮ

В главе IX (см. § 9.3) мы доказали, что всякий КС-язык допускается некоторым МП-автоматом. Наметим теперь доказательство обратной теоремы.

15.3.1. Теорема

Всякий язык, допускаемый МП-автоматом, есть КС-язык. Идея доказательства состоит в следующем. Пусть некоторый МП-автомат $\mathfrak{A}$ допускает язык L. Запишем процесс работы автомата $\mathfrak{A}$ в виде цепочки, состоящей из символов, сопоставленных командам автомата, выполняемым на последовательных шагах процесса; обозначим через L' множество всех таких цепочек Нетрудно видеть, что по автомату $\mathfrak{A}$ можно построить односторонний конечный преобразователь, отображающий L' на L (этот преобразователь должен переводить каждый символ команды в читаемый при выполнении этой команды входной символ). Поэтому доста-

точно показать, что L' есть КС-язык. Но цепочки языка L' устроены весьма сходно с цепочками языка Дика (и даже ограниченного языка Дика): команды записи на рабочей ленте соответствуют «левым скобкам», команды чтения на рабочей ленте — «правым»; правда, одной и той же «левой скобке» могут отвечать «правые скобки» различных типов (и обратно), поскольку на входной ленте могут читаться разные символы. Однако можно построить такой односторонний конечный преобразователь τ, что прообразы цепочек языка L' относительно осуществляемого им преобразования будут принадлежать некоторому языку Дика D^*, и $\tau^{-1}(L')$ будет пересечением D^* с некоторым стандартным А-языком, разрешенные диграммы которого соответствуют, грубо говоря, тем парам команд автомата $\mathfrak{A}$, которые можно выполнять непосредственно друг за другом.

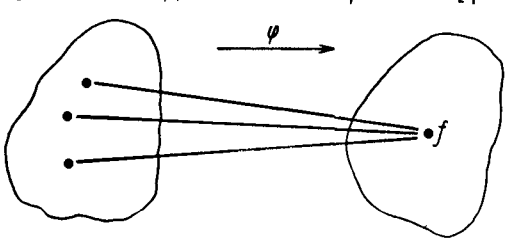
Из этой теоремы немедленно следует другая, о которой мы говорили в п. 9 3.2. Принимая во внимание ее важность, сформулируем ее еще раз.

Теорема. *Класс языков, допускаемых (порождаемых) автоматами с магазинной памятью, совпадает с классом контекстно-свободных языков.*

15.3.2. Замечание о неоднозначности КС-языков

Рассмотрим гомоморфное отображение φ произвольного стандартного КС-языка C_S на произвольный КС-язык. Язык C_S, цепочкам которого приписывает определенную структуру его «естественная» грамматика, однозначен; произвольный КС-язык, который порождается грамматикой, полученной с помощью тройки (D^*, Q, φ), где D^* — язык Дика и Q — стандартный А-язык, может быть и неоднозначным. Степень неоднозначности цепочки f равна мощности множества ее прообразов относительно φ, т. е.

$$\text{степень неоднозначности } f = \operatorname{card}[\varphi^{-1}(f)].$$



§ 15.4. УПРАЖНЕНИЯ

1. Пусть имеется язык Дика в алфавите $\{a, b, \ldots; a', b', \ldots\}$ и стандартный А-язык, заданный системой уравнений, как описано в п. 14.1.2. Сколькими уравнениями будет задаваться соответ-

ствующий КС-язык, если воспользоваться для их вывода процедурой, указанной в п. 15.2 3?

2. Стандартный КС-язык называется *ограниченным*, если он удовлетворяет следующим двум дополнительным условиям:

1°. Для всякого разрешенного перехода от буквы без штриха к другой букве без штриха разрешается переход в обратном направлении между соответствующими штрихованными буквами, и обратно; например, если есть ab, то должно существовать $b'a'$, а если есть $a'b'$, то должно существовать и ba.

2°. Разрешенными переходами типа xy' и $x'y$ могут быть только переходы типа xx', соответственно $x'x$.

Первый вопрос. Построим матрицу разрешенных переходов, выписывая левые члены диграмм (т. е. строки матрицы) в порядке $a, b, c, \ldots, a', b', c', \ldots$, а правые (т. е. столбцы) — в порядке $a', b', c', \ldots, a, b, c, \ldots$. В этой матрице естественным об разом выделяется четыре «блока»:

$$\mathfrak{A}\mathfrak{A}'; \quad \mathfrak{A}'\mathfrak{A}'; \quad \mathfrak{A}\mathfrak{A}; \quad \mathfrak{A}'\mathfrak{A}.$$

Как сформулировать в терминах такой матрицы условия 1° и 2°?

Второй вопрос. Исследовать, как влияет на природу ограниченного стандартного КС-языка C_S введение следующих дополнительных ограничений на переходы типа $x'y$ и xy':

а) Переходы типа $x'y$ или xy' запрещены. Показать, что тогда C_S является линейным.

б) Число переходов типа $x'y$ и xy' не превышает некоторой постоянной. Показать, что тогда C_S является металинейным.

Третий вопрос. Дано множество начальных букв и матрица разрешенных диграмм; построить соответствующий язык C_S.

3. Всякий ли стандартный КС-язык является детерминированным?

4. КС-языки можно классифицировать по двум признакам: однозначные/неоднозначные, детерминированные/недетерминированные, так что получается четыре класса Все ли эти классы непусты? Привести примеры

5. Рассмотрим язык L над $\{a, b\}^*$:

$$L = \{xcy \mid\mid x \mid = \mid y \mid \& \, x \neq \tilde{y}\}.$$

Показать, что дополнение этого языка до $\{a, b\}^*$ есть КС-язык. Показать, что $L_1 = \{xcy \mid x \neq y\}$ есть КС-язык.

6. Провести выкладки, намеченные в п. 15.2.3 (доказательство основной теоремы о КС-языках), для случая двойного зеркального языка (см. п. 7.4.3).

Глава XVI

АЛГЕБРАИЧЕСКИЕ ЯЗЫКИ

§ 16.1. ДОПОЛНИТЕЛЬНЫЕ СВЕДЕНИЯ О ФОРМАЛЬНЫХ СТЕПЕННЫХ РЯДАХ

В гл. XI мы ввели понятие формального (степенного) ряда, членами которого являются ассоциативные, но не коммутативные одночлены. Там же были указаны (в основном на интуитивном уровне) некоторые приложения формальных рядов к теории КС-языков. В этой главе мы займемся более строгим и детальным изучением формальных рядов, имея в виду другие приложения.

Мы будем излагать теорию формальных рядов в общем виде, предполагая при этом, что область Ω коэффициентов является кольцом. Читатель сам сможет сделать изменения, необходимые в тех случаях, когда Ω — не кольцо, а полукольцо. Наиболее интересные приложения получаются при $\Omega = \mathbf{Z}$ (кольцо целых чисел) и $\Omega = \mathbf{N}$ (полукольцо натуральных чисел).

Мы старались излагать материал данной главы как можно более просто; тем не менее нам приходится предполагать, что читатель знаком хотя бы с элементами того, что называют «современной математикой». В особенности желательно, чтобы он имел представление о формальной теории многочленов от одной или нескольких неизвестных. Читателя, не обладающего соответствующими познаниями, мы должны предостеречь по крайней мере от одной распространенной ошибки: от смешения многочлена с *полиномиальной функцией*.

В многочлене, таком, например, как $3x^2 + 2x + 1$, буква x вовсе не обязательно обозначает переменную, принимающую значения на том или ином множестве чисел: x может представлять собой неизвестную или, если угодно, многочлен, состоящий из одного члена x. Многочлен может описывать не только определенную полиномиальную функцию, но и другие объекты.

16.1.1. Ассоциативные многочлены

Пусть $X = \{x_i \mid 1 \leqslant i \leqslant n\}$ — некоторый алфавит; образуем свободную полугруппу X^*. Каждой цепочке $x_{i_1} \ldots x_{i_r}$, принадлежащей X^*, можно сопоставить *длину по* x_1 — число вхождений x_1, *длину по* x_2 и т. д., а также *общую длину* (называемую просто длиной цепочки) — сумму длин по всем x_i. Пустая цепочка имеет длину 0.

Пусть, далее, имеется ассоциативное и коммутативное кольцо с единицей $\Omega = \{\alpha, \beta, \ldots\}$. Тогда мы можем построить *свободную ассоциативную алгебру* $\Omega[X^*]$ следующим образом.

Для любых $\alpha \in \Omega$ и $f \in X^*$ выражение αf есть по определению *ассоциативный одночлен* с коэффициентом α. *Произведением* двух одночленов αf и βg (взятых в этом порядке) является выражение $(\alpha\beta)fg$, где произведение $\alpha\beta$ берется в смысле Ω, а fg — в смысле X^* (иначе говоря, fg есть конкатенация цепочек f и g).

Элементами алгебры $\Omega[X^*]$ являются *ассоциативные многочлены* от x_i с коэффициентами из Ω. Такой многочлен имеет вид

$$P = \alpha_1 f_1 + \ldots + \alpha_k f_k, \quad \text{где} \quad \alpha_i \in \Omega, \quad f_i \in X^*,$$

и содержит конечное число членов с ненулевыми коэффициентами; можно также вводить члены вида $0f$, где 0 обозначает нулевой элемент кольца Ω.

Сумма двух многочленов определяется по обычному правилу: если многочлены P и Q содержат соответственно *подобные* члены αf и βf, то многочлен $R + Q$ содержит член $(\alpha + \beta)f$, где сумма $\alpha + \beta$ строится по правилу кольца Ω. Многочлен $P + Q$ не имеет членов, отличных от тех, которые были получены по указанному правилу; поскольку можно вводить члены вида $0f$, это правило оказывается всегда применимым.

Заметим, что всякий многочлен можно трактовать как сумму его членов, рассматриваемых как многочлены, состоящие из одного члена каждый. Тем самым содержательно оправдывается употребление знака + в записи многочлена.

Умножение многочленов определяется следующим правилом:

$$\left(\sum_{i=1}^{r} (\alpha_i f_i)\right)\left(\sum_{j=1}^{s} (\beta_j g_j)\right) = \sum_{i=1,\ j=1}^{r,\ s} (\alpha_i \beta_j)(f_i g_j).$$

Эта операция ассоциативна, но не коммутативна.

Ясно, что $\Omega[X^*]$ есть кольцо; его нулевой элемент — это нулевой многочлен (т. е. многочлен, все коэффициенты которого равны нулю); его единичным элементом является $1e$, где 1 — единичный элемент кольца Ω.

16.1.2. Композиция многочленов

Пусть Y — алфавит; возможно, что $Y = X$. Пусть, далее, $f \in \Omega[X^*]$, $g_i \in \Omega[Y^*]$, $i = 1, \ldots, n$; $n = \operatorname{card}(X)$. Поскольку мы уже определили сложение и умножение многочленов, можно теперь естественным образом определить *композицию* многочленов:

$$f(g_1, \ldots, g_n)\ ^{1}).$$

Пример. Пусть $f(x_1, x_2) = x_1 x_2$; тогда $f(g_1, g_2) = g_1 g_2$. Отсюда видно, что умножение может рассматриваться как частный случай композиции.

---

[1]) Т. е. результат подстановки в f многочленов $g_1, \ldots, g_n$ вместо $x_1, \ldots, x_n$ соответственно. — *Прим. ред.*

Пусть буквы $A_1, \ldots, A_n$, образующие алфавит $\mathfrak{A}$, обозначают переменные, принимающие значения в $\Omega[T^*]$, где $T = \{t_1, \ldots, t_m\}$.

Пусть, кроме того, имеется еще отображение σ алгебры $\Omega[\mathfrak{A}^*]$ в $\Omega[(\mathfrak{A} \cup T)^*]$, определяемое условиями

$$\sigma\colon A_j \to g_j(t_1, \ldots, t_m, A_1, \ldots, A_n), \quad j = 1, \ldots, n.$$

Это отображение индуцирует некоторое отображение $\Omega^n[T^*]$ в себя следующим образом: n-ке многочленов $[s_1, \ldots, s_n]$ ставится в соответствие n-ка многочленов $[g_1(t_1, \ldots, t_m, s_1, \ldots, s_n), \ldots, g_n(t_1, \ldots, t_m, s_1, \ldots, s_n)]$.

Пример.

$$\mathfrak{A} = \{A, B\}, \quad T = \{a, b\},$$
$$A \to a + AbB + BbA,$$
$$B \to AbA + BbB.$$

Паре многочленов $(0, 0)$ соответствует здесь пара $(a, 0)$, паре (b, a) — пара $(a + b^2a + ab^2, b^3 + aba)$ и т. д.

16.1.3. Формальные ряды

Построим теперь *расширенную алгебру* $\Omega[[X^*]]$, содержащую $\Omega[X^*]$ в качестве подалгебры. Элементами этой алгебры будут *формальные ряды*

$$\alpha_1 f_1 + \ldots + \alpha_n f_n + \ldots,$$

которые могут содержать неограниченное число неподобных членов с ненулевыми коэффициентами. *Сложение рядов* определяется так же, как и сложение многочленов.

Чтобы определить *умножение рядов*, будем упорядочивать их по возрастанию степеней (длин) цепочек и представлять каждую цепочку всеми способами в виде произведения (конкатенации) двух сомножителей. Цепочка $h = x_{i_1} x_{i_2} \ldots x_{i_k}$ представляется либо в виде eh, либо в виде $\left(x_{i_1}\right)\left(x_{i_2} \ldots x_{i_k}\right)$ и т. д.

Цепочка h в произведении двух рядов будет иметь коэффициент $\sum \alpha_i \beta_j$, где сумма берется по всем возможным разбиениям цепочки h на два сомножителя, первый из которых — член первого ряда, а второй — член второго ряда. Таким образом, для того чтобы определить коэффициент цепочки h в произведении двух рядов, требуется перебрать *конечное* (и заранее известное) число их членов. Следовательно, произведение рядов определяется эффективно.

Легко видеть, что $\Omega[[X^*]]$ — кольцо. Эта расширенная алгебра содержит подмножество, изоморфное кольцу $\Omega[X^*]$ [1]), и мы будем

[1]) Это подмножество состоит из тех рядов, у которых лишь конечное число коэффициентов отлично от нуля. — *Прим. ред.*

отождествлять это подмножество с самим $\Omega[X^*]$, что равносильно трактовке многочлена как «вырожденного ряда».

16.1.4. Норма пространства формальных рядов

Назовем *порядком* формального ряда наименьшую из длин членов ряда — непустых цепочек с ненулевыми коэффициентами, остающихся после приведения подобных членов. Порядок нулевого ряда (т. е. такого, у которого все коэффициенты равны нулю) будем считать по определению равным $+\infty$.

Пример. Пусть $X = \{x, y\}$; ряды выписываются по возрастающим степеням.

Ряд

$$e + xy + x^2y^2 + \ldots + x^ny^n + \ldots$$

содержит пустую цепочку e, и его порядок равен нулю.

Ряд

$$xx + yy + xyx + yxy + \ldots + xy^kx + yx^ky + \ldots$$

имеет порядок 2 (длина, или степень цепочек xx и yy).

Сложение. При сложении двух рядов возможны следующие случаи:

— порядок одного ряда строго меньше порядка другого; тогда порядок суммы равен меньшему из этих двух порядков;

— оба ряда имеют одинаковый порядок; тогда, если члены низших степеней не уничтожаются, порядок суммы остается таким же, а если члены низших степеней взаимно уничтожаются, порядок суммы увеличивается.

Случай, когда хотя бы один из рядов оказывается нулевым, не ведет к дополнительному усложнению.

В результате мы получаем следующую лемму:

Лемма 1. *Порядок суммы рядов* (s) *и* (s') *удовлетворяет неравенству*

$$\operatorname{ord}[(s) + (s')] \geqslant \min[\operatorname{ord}[(s)], \operatorname{ord}[(s')]]\ ^1).$$

Умножение. При перемножении двух рядов их члены, имеющие минимальные степени, дают член произведения, имеющий минимальную степень, если только Ω не содержит делителей нуля. В противном случае суммарная степень произведения может оказаться больше суммы степеней сомножителей. Это дает такую лемму:

Лемма 2. *Порядок произведения рядов* (s) *и* (s') *удовлетворяет неравенству*

$$\operatorname{ord}[(s)(s')] \geqslant \operatorname{ord}[(s)] + \operatorname{ord}[(s')];$$

---

[1]) ord означает порядок. — *Прим. ред.*

в случае, когда кольцо Ω является областью целостности, неравенство можно заменить равенством.

Норма в кольце $\Omega[[X^*]]$. Возьмем любое действительное число, строго большее единицы, например 2. Определим ***норму*** ряда (s) — обозначение $\mathrm{val}[(s)]$ — следующим образом:

$$\mathrm{val}[(s)] = 2^{-\mathrm{ord}[(s)]}.$$

Примеры. Для рядов, фигурировавших в предыдущих примерах, получим соответственно

$$2^0 = 1 \quad \text{и} \quad 2^{-2} = \frac{1}{4}.$$

В силу нашего определения для нулевого ряда (s_0) имеем

$$\mathrm{val}[(s_0)] = 2^{-\infty} = 0.$$

Сложим ряды (s) и (s'); тогда

$$\mathrm{val}[(s) + (s')] = 2^{-\mathrm{ord}[(s)+(s')]};$$

но поскольку

$$\mathrm{ord}[(s) + (s')] \geqslant \min[\mathrm{ord}[(s)],\ \mathrm{ord}[(s')]],$$

мы получаем следующую лемму:

Лемма 3.

$$\mathrm{val}[(s) + (s')] \leqslant \max[\mathrm{val}[(s)],\ \mathrm{val}[(s')]].$$

Примеры. Пусть ряд (s) начинается с x и ряд (s') начинается с xy. Тогда ряд $(s) + (s')$ начинается с x. Имеем

$$\mathrm{val}[(s)] = 2^{-1}, \quad \mathrm{val}[(s')] = 2^{-2},$$

$$\mathrm{val}[(s) + (s')] = 2^{-1}.$$

Если ряд (s) начинается с $x + xy + \ldots$, а ряд (s') начинается с $-x + xy + \ldots$, то

$$\mathrm{val}[(s)] = 2^{-1}, \quad \mathrm{val}[(s')] = 2^{-1},$$

$$\mathrm{val}[(s) + (s')] = 2^{-2}.$$

Из леммы 2 вытекает также

Лемма 4. *В кольце* $\Omega[[X^*]]$

$$\mathrm{val}[(s)(s')] \leqslant \mathrm{val}[(s)] \cdot \mathrm{val}[(s')].$$

Читатель докажет сам (упр. 1 из § 16.4), что функция $\mathrm{val}[(s)]$ является нормой для кольца $\Omega[[X^*]]$.

16.1.5. Расстояние между рядами

Пусть даны два ряда (s) и (s'). *Расстоянием* между этими рядами (обозначение: $d[(s), (s')]$) мы назовем число

$$\mathrm{val}[(s) - (s')].$$

Это расстояние обладает следующими свойствами:

I. $d[(s), (s)] = 0$.

В самом деле, $d[(s), (s)] = \mathrm{val}[(s) - (s)] = \mathrm{val}[s_0] = 0$.

II. $d[(s), (s')] = d[(s'), (s)]$.

III. $d[(s), (s'')] \leqslant \max[d[(s), (s')], d[(s'), (s'')]]$.

Действительно,

$$(s) - (s'') = [(s) - (s')] + [(s') - (s'')];$$

остается применить лемму 3.

Заметим, что свойство III сильнее, чем неравенство треугольника:

IIIa. $d[(s), (s'')] \leqslant d[(s), (s')] + d[(s), (s'')]$.

Свойства I, II и IIIa — обычные аксиомы расстояния; это оправдывает название, данное нами функции d. Пространство, в котором определено расстояние, удовлетворяющее этим аксиомам, называют *метрическим*; если же кроме IIIa выполняется более сильное неравенство III, то пространство называют *ультраметрическим*.

Поскольку расстояние между двумя рядами не может быть больше 1, пространство рядов *ограничено*: оно имеет диаметр 1.

Итак, имеет место

Предложение. *Пространство формальных рядов, в котором указанным выше способом введено расстояние, является ультраметрическим.*

16.1.6. Предел последовательности рядов

Введя понятие расстояния, мы можем определить предел последовательности рядов. Пусть имеется последовательность формальных рядов $(s_1), \ldots, (s_p), \ldots$; пусть (s) — фиксированный ряд

Если $d[(s_p) - (s)]$ стремится к нулю, говорят, что последовательность $(s_1), \ldots, (s_p), \ldots$ *стремится к* (s) (при p, стремящемся к бесконечности) или, иначе, что (s) есть *предел* этой последовательности.

Обозначим через P_0, P_1, ..., P_r, ... последовательность однородных многочленов, такую, что степень многочлена P_r равна r. Последовательность

$$P_0,$$
$$P_0 + P_1,$$
$$\ldots\ldots$$
$$P_0 + P_1 + \ldots + P_r,$$
$$\ldots\ldots\ldots$$

есть последовательность многочленов и, следовательно, последовательность рядов, стремящаяся (в определенном выше смысле) к ряду

$$P_0 + P_1 + \ldots + P_r + \ldots.$$

Пусть (s_1), (s_2), ... — произвольные ряды порядков 1, 2, ..., соответственно. Последовательность

$$(t_0) = P_0 + (s_1),$$
$$(t_1) = P_0 + P_1 + (s_2),$$
$$\ldots\ldots\ldots$$
$$(t_r) = P_0 + P_1 + \ldots + P_r + (s_{r+1})$$

стремится к тому же ряду.

Будем говорить, что два ряда *r-эквивалентны*, если порядок их разности больше или равен $r + 1$. Можно сформулировать следующее

Предложение. *Если в сходящейся последовательности рядов* (t_0), ..., (t_r), ... *заменить каждый ряд* (t_r) *произвольным r-эквивалентным ему рядом, то предел последовательности не изменится.*

16.1.7. Полнота пространства $\Omega[[X^*]]$

Пусть (s_1), (s_2), ..., (s_n), ... — бесконечная последовательность элементов в метрическом пространстве. *Критерием Коши* называется следующее условие:

$$(\forall \varepsilon > 0)\ (\exists N)\ [p > N \,\&\, q > N \Rightarrow d[(s_p), (s_q)] < \varepsilon].$$

Всякая сходящаяся последовательность удовлетворяет критерию Коши. Пространство, в котором всякая последовательность, удовлетворяющая критерию Коши (*последовательность Коши*), сходится, называется *полным*.

Покажем, что пространство $\Omega[[X^*]]$ является полным.

Пусть (s_n) — некоторая последовательность Коши. Тогда можно найти номер n_1, начиная с которого все члены последовательности 1-эквивалентны, затем номер n_2, начиная с которого все члены 2-эквивалентны, и т. д.

Теперь легко получить последовательность многочленов, сходящуюся к некоторому ряду (s), и показать, что последовательность (s_n) сходится к (s) [1]).

16.1.8. Квазирегулярность. Итерация

Назовем ряд *квазирегулярным*, если в нем коэффициент при пустой цепочке равен нулю. В частности, нулевой ряд является квазирегулярным.

Пусть (s) — квазирегулярный ряд; наименьшая из возможных степеней его членов не меньше единицы. Образуем ряд $(s)^2$; наименьшая степень его членов не меньше двух. Аналогично для $(s)^3$ и т. д. При $p \to \infty$ порядок ряда $(s)^p$ стремится к бесконечности.

Пусть (s_0) — нулевой ряд. Из только что сказанного следует, что $d[(s)^p, (s_0)]$ стремится к нулю; стало быть, $(s)^p$ стремится к нулевому ряду.

Рассмотрим последовательность

$$(t_1) = (s),$$
$$(t_2) = (s) + (s)^2,$$
$$(t_3) = (s) + (s)^2 + (s)^3,$$
$$\cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot$$

Эта последовательность удовлетворяет критерию Коши и, следовательно, сходится к пределу, который мы обозначим через $(s)^*$.

Для ряда $(s)^*$ имеют место следующие очевидные соотношения:

$$(s) + (s)^2 + \ldots + (s)^p + \ldots = (s)^*,$$
$$(s) + (s)^* (s) = (s) + (s)(s)^* = (s)^*.$$

Положим

$$(s') = e - (s); \quad (s'') = e + (s)^*.$$

Тогда

$$(s')(s'') = e - (s) - (s)(s)^* + (s)^* = e.$$

Обратно, пусть (s') — ряд, в котором коэффициент при пустой цепочке равен единице кольца Ω. Положим $(s) = e - (s')$, построим $(s)^*$, как и выше, а затем положим $(s'') = e + (s)^*$.

Ряд (s'') будет *обратным* для ряда (s'). Если Ω — тело, то обращение возможно для всякого ряда, у которого коэффициент

---

[1]) В самом деле, r-эквивалентность рядов означает совпадение всех их членов, являющихся цепочками длины $<r$; если обозначить через t_r многочлен, состоящий из всех тех членов рядов $s_{n_r}, s_{n_{r+1}}, \ldots$, степени которых $< r$ (у всех этих рядов такие члены одинаковы), и положить $(s) = \lim\limits_{n\to\infty} (t_n)$, то $(s) = \lim\limits_{n\to\infty} (s_n)$. — *Прим. ред.*

при пустой цепочке отличен от нуля. В этом случае, в частности, всякий многочлен $Q(X)$ из $\Omega[X^*]$, такой, что $Q(0) \neq 0$, обратим в $\Omega[[X^*]]$. Отсюда следует, что кольцо $\Omega[[X^*]]$ содержит все частные вида

$$P(X) \cdot \frac{1}{Q(X)} \quad \text{и} \quad \frac{1}{Q(X)} \cdot P(X), \quad \text{где} \quad Q(0) \neq 0,$$

образующие подмножество множества так называемых *рациональных рядов.*

Ряд $(s)^*$ называют *квазиобратным* для ряда (s).

16.1.9. Подстановки в формальных рядах

Пусть имеются алфавиты X и Y и ряд $(t) \in \Omega[[Y^*]]$. Когда можно подставить в ряд (t) вместо каждого y_j некоторый ряд $(s_j) \in \Omega[[X^*]]$?

Это возможно, в частности, если (t) — многочлен. Если (t) — ряд, то это возможно при условии, что ряды (s_j) квазирегулярны (убедиться в этом предоставляется читателю в качестве упражнения).

Пример. Построение ряда $(s)^*$ сводится к подстановке ряда (s) в ряд

$$1 \cdot Y + 1 \cdot Y^2 + \ldots + 1 \cdot Y^n + \ldots.$$

16.1.10. n-ки формальных рядов

Пусть n — целое положительное число. Упорядоченные n-ки формальных рядов, принадлежащих $\Omega[[X^*]]$, образуют пространство, в котором за расстояние между двумя n-ками можно принять сумму расстояний между их компонентами.

Поскольку многочлен — частный случай формального ряда, то факты, установленные при изучении формальных рядов, могут пролить свет на вопросы, относящиеся к композиции многочленов.

Вернемся к ситуации, описанной в п. 16.1.2. Для простоты мы будем говорить о парах рядов или многочленов, что не уменьшает общности рассуждений.

Пусть A и B — переменные, принимающие значения из $\Omega[X^*]$; пусть, кроме того, имеется отображение

$$A \to \sigma(x_1, \ldots, x_m, A, B),$$
$$B \to \tau(x_1, \ldots, x_m, A, B).$$

Это отображение индуцирует отображение множества $\Omega^2[X^*]$ в себя: точка $M(P, Q)$ переходит в точку

$$M(P', Q') = (\sigma(x_1, \ldots, x_m, P, Q), \tau(x_1, \ldots, x_m, P, Q)).$$

Посмотрим, что произойдет при этом с расстоянием. Возьмем точки (P_1, Q_1) и (P_2, Q_2). Расстояние между ними равно $d(P_1, P_2) + d(Q_1, Q_2)$. Расстояние между их образами равно $d(P_1', P_2') + d(Q_1', Q_2')$.

Однако

$$d(P_1, P_2) = \mathrm{val}(P_2 - P_1),$$
$$d(P_1', P_2') = \mathrm{val}(P_2' - P_1').$$

В то же время

$$P_2' - P_1' = \tau(x_1, \ldots, x_m, P_2, Q_2) - \sigma(x_1, \ldots, x_m, P_1, Q_1).$$

В этой разности члены, не содержащие ни P, ни Q, взаимно уничтожаются. Предположим, что в многочленах σ и τ коэффициенты при цепочках e, A и B равны нулю («условие повышения степени»). Тогда нетрудно видеть, что порядок ряда

$$\sigma(x_1, \ldots, x_m, P_2, Q_2) - \sigma(x_1, \ldots, x_m, P_1, Q_1)$$

будет больше минимума порядков рядов $P_2 - Q_2$ и $P_1 - Q_1$ на некоторое целое число, зависящее только от σ.

Аналогичное утверждение справедливо для разности $Q_2' - Q_1'$.

Поэтому расстояние между M_1' и M_2' получается из расстояния между M_1 и M_2 умножением на число λ, строго меньшее единицы и зависящее только от σ и τ:

$$d(M_1', M_2') = \lambda d(M_1, M_2), \qquad 0 < \lambda < 1.$$

Таким образом, рассматриваемое отображение является *стягивающим.*

Мы воспользуемся этим результатом в п. 16.2.1.

16.1.11. Формальные ряды с коммутативными переменными

Ясно, что можно было бы построить аналогичную теорию, допустив коммутативность образующих; тогда вместо X^* мы рассматривали бы свободную абелеву полугруппу с единицей над алфавитом X.

Всякому вычислению в $\Omega[[X^*]]$ соответствует вычисление, полученное в предположении коммутативности. Точнее, существует такой гомоморфизм кольца $\Omega[[X^*]]$, при котором всякая цепочка $f \in X^*$ отображается в соответствующий элемент свободной абелевой полугруппы.

Пример. Пусть Ω — кольцо целых чисел. Рассмотрим многочлен

$$f(x, y, a) = a + xy^2$$

и будем итерировать его следующим образом:

$$Y_0 = 0e, \ldots, Y_n = f(x, Y_{n-1}, a).$$

При некоммутативных неизвестных имеем

$$Y_0 = 0e,$$
$$Y_1 = a,$$
$$Y_2 = a + xa^2,$$
$$Y_3 = a + x(a + xa^2)^2 = a + xa^2 + xaxa^2 + x^2a^3 + x^2a^2xa^2,$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

При коммутативных неизвестных получаем

$$\overline{Y}_0 = 0e,$$
$$\overline{Y}_1 = a,$$
$$\overline{Y}_2 = a + xa^2,$$
$$\overline{Y}_3 = a + a^2x + 2a^3x^2 + a^4x^3,$$
$$\cdots\cdots\cdots\cdots$$

§ 16.2. АЛГЕБРАИЧЕСКИЕ РЯДЫ

16.2.0. О термине «алгебраический»

Пусть K — поле и $K[x]$ — кольцо коммутативных многочленов над K. Рассмотрим уравнение

$$a_0x^n + \ldots + a_n = 0,$$

не имеющее решения в K.

Обозначим через θ новый элемент, удовлетворяющий данному уравнению; в этом случае $K[\theta]$ называется *алгебраическим расширением* поля K.

Пример. Рассмотрим поле рациональных чисел K и уравнение $x^2 - 2 = 0$; можно ввести символ θ, такой, что $\theta^2 - 2 = 0$. Отсюда следует, что

$$x^2 - 2 = x^2 - \theta^2 = (x + \theta)(x - \theta).$$

Таким образом, можно найти и второй корень: он равен $-\theta$.

Если K — числовое поле, допускающее погружение в $\mathbf{R}$ (поле действительных чисел) или в $\mathbf{C}$ (поле комплексных чисел), то θ можно отождествить с некоторым действительным или комплексным числом.

Пример. В предыдущем примере $\theta = \sqrt{2} = 1{,}41421\ldots$.

Числа, являющиеся корнями алгебраических уравнений (т. е. уравнений вида $f(x) = 0$, где f — многочлен) с рациональными коэффициентами, называются *алгебраическими*.

Можно доказать, например, что число π не является алгебраическим.

В том же самом смысле термин *алгебраический* используется и применительно к полю функций.

Пример. Пусть $Q(x)$ — поле рациональных дробей с рациональными коэффициентами. Расширим это поле с помощью функции y, такой, что

$$y = xy^2 + 1;$$

эта функция является алгебраической.

Формальный ряд не следует смешивать с функцией, которую можно связать с этим рядом (ср. начало § 16.1). Рядом можно воспользоваться для представления той или иной функции, если только мы умеем сопоставить данному ряду его сумму. В классическом анализе пользуются «обычной» суммой, определенной в случае, когда ряд сходится; тогда можно найти радиус сходимости и т. д.

Мы, однако, останемся на чисто формальной точке зрения. При таком подходе естественно говорить, что формальный ряд, удовлетворяющий (конечному) алгебраическому уравнению, является алгебраическим.

Наконец, мы будем употреблять слово *алгебраический* в том же самом смысле и для некоммутативных рядов.

16.2.1. Системы уравнений специального вида

Пусть $X = \{x_i \mid 1 \leqslant i \leqslant m\}$ — базовый алфавит, а $\mathfrak{A} = \{A_j \mid 1 \leqslant j \leqslant n\}$ — алфавит, символами которого обозначаются многочлены или формальные ряды из $\Omega[[X^*]]$. Пусть, далее, σ_j — многочлены над $\Omega[\{\mathfrak{A} \cup X\}^*]$.

Будем рассматривать системы уравнений вида $A_j = \sigma_j$, $j = 1, \ldots, n$; решения их мы будем искать в $\Omega^n[[X^*]]$.

Системы уравнений, рассматривавшиеся в гл. XI, являются по существу частным случаем таких систем (обоснование этого утверждения будет дано ниже).

Будем считать, что многочлены σ_j удовлетворяют «условию повышения степени» (см. выше, п. 16.1.10).

Указанные системы можно решать методом последовательных приближений. Этот метод основан на следующей теореме:

Теорема. *Пусть E — полное метрическое пространство и f — стягивающее отображение E в себя. Тогда уравнение $x = f(x)$ имеет точно одно решение a и для произвольной точки $x_0 \in E$ последовательность $x_0, f(x_0), f(f(x_0)), \ldots$ сходится к a.*

Доказательство этой теоремы для общего случая предоставляется читателю в качестве упражнения. В том частном случае, который нас здесь интересует, а именно в случае ограниченного пространства, достаточно заметить, что последовательность $f^n(E)$ является убывающей и диаметр множества $f^n(E)$ не превосходит λ^n, где λ имеет тот же смысл, что и в п. 16.1.10.

Замечание. В примерах гл. XI мы брали в качестве x_0 аннулятор, которому в обозначениях данной главы соответствует $0e$. Мы поступали так из соображений удобства. В действительности значение x_0 можно выбрать произвольно, однако в этом случае нам, возможно, придется иметь дело с «лишними» выражениями.

Пример. Вернемся к «бесскобочной» грамматике $S = aSS + b$, рассмотренной в гл. XI, однако возьмем другое начальное значение, например:

$$S_0 = a$$
$$S_1 = a^3 + b$$
$$S_2 = a(a^3 + b)(a^3 + b) + b =$$
$$= a^7 + a^4b + aba^3 + ab^2 + b.$$

Как видим, появляется «утяжеленная» последовательность S_r, где S_r r-эквивалентно «хорошему» значению (получаемому при $S_0 = 0$).

Теорема. *Для алгебраической системы уравнений*

$$A_j = \sigma_j, \quad j = 1, \ldots, n,$$

где во всех многочленах σ_j коэффициенты при e, $A_1, \ldots, A_n$ равны нулю, существует ровно одна n-ка рядов из $\Omega[[X^*]]$, которая удовлетворяет этой системе (и которую можно получить путем последовательных приближений).

§ 16.3. ПРИЛОЖЕНИЯ К ЯЗЫКАМ

Возьмем в качестве Ω кольцо целых чисел $\mathbf{Z}$. Некоторые примеры приложений систем уравнений к языкам были приведены в гл. XI. Сейчас мы укажем интерпретацию в терминах языков некоторых результатов этой главы.

16.3.1. Вычитание языков

Рассмотрим некоммутативное уравнение

$$S = a - SbS \tag{F}$$

и следующие приближения:

$$S^{(0)} = a,$$
$$S^{(1)} = a - S^{(0)}bS^{(0)} = a - aba,$$
$$S^{(2)} = a - S^{(1)}bS^{(1)} = a - (a - aba)b(a - aba) =$$
$$= a - aba + 2ababa - abababa,$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

Формальный ряд, являющийся решением уравнения (F), имеет как положительные, так и отрицательные коэффициенты.

Положим $S = S^+ - S^-$ и подставим это выражение в (F). Имеем

$$S^+ - S^- = a - (S^+ - S^-)\,b\,(S^+ - S^-),$$

откуда

$$S^+ - S^- = (a + S^+bS^- + S^-bS^+) - (S^+bS^+ + S^-bS^-).$$

Рассмотрим теперь систему

$$\text{(F}'\text{)} \qquad \begin{aligned} S^+ &= a + S^+bS^- + S^-bS^+, \\ S^- &= S^+bS^+ + S^-bS^-. \end{aligned}$$

Уравнения системы (F′) имеют положительные коэффициенты, и (F′) допускает в качестве решения пару рядов с положительными коэффициентами; обозначим эту пару через (σ^+, σ^-). Тогда, если σ — формальный ряд, являющийся решением уравнения (F), то

$$\sigma = \sigma^+ - \sigma^-.$$

Системе (F′) мы можем поставить в соответствие две грамматики: G^+ с аксиомой S^+ и G^- с аксиомой S^-. На этом примере мы разъясним понятие вычитания языков.

Рассмотрим две КС-грамматики G^+ и G^-, порождающие цепочки с определенными степенями неоднозначности. Будем считать, что «грамматика» $G^+ - G^-$ порождает цепочки как с положительными, так и с отрицательными степенями неоднозначности, причем с положительными степенями неоднозначности в «язык», порождаемый грамматикой $G^+ - G^-$ (иначе — в разность языков), войдут цепочки из $L(G^+)$, с отрицательными — из $L(G^-)$; если цепочка f порождается обеими грамматиками, то ее степень неоднозначности относительно $G^+ - G^-$ равна $\alpha - \beta$, где α и β — степени неоднозначности f относительно G^+ и G^- соответственно.

Разложение, аналогичное построенному в этом примере, можно произвести и в общем случае: всякой системе уравнений с целыми — положительными и отрицательными — коэффициентами можно поставить в соответствие две системы уравнений с положительными коэффициентами так, что данная система будет соответствовать их разности.

Пример. Пусть язык

$$I_m = \{a^n c a^n \mid n \geqslant 0\}$$

порождается уравнением

$$S = aSa + c.$$

Обозначим через L'_m язык, порождаемый уравнением

$$(G):\ S = aSa + S - a^k c a^k, \quad \text{где} \quad k - \text{постоянная}.$$

Язык L'_m можно получить как разность языков, задаваемых уравнениями

$$(G^+):\ S^+ = aS^+a + c,$$

$$(G^-):\ S^- = aS^-a + a^k c a^k,$$

$$L(G^+):\ \{a^n c a^n \mid n \geqslant 0\},$$

$$L(G^-):\ \{a^n c a^n \mid n \geqslant k\},$$

$$L'_m = L(G^+) - L(G^-) = \{a^n c a^n \mid 0 \leqslant n < k\}.$$

Язык L'_m конечен; применяя к (G) метод последовательных приближений, можно видеть, что все цепочки, содержащие вхождение подцепочки $a^k c a^k$, в конце концов исчезают.

16.3.2. Рациональные языки

В п. 16.1.8 мы видели, что кольцо $\mathbf{Z}[[X^*]]$ содержит всевозможные частные вида $\frac{P(x_1, \ldots, x_m)}{Q(x_1, \ldots, x_m)}$, где P, Q — многочлены, а в Q коэффициент при пустой цепочке равен 1 (или -1).

Опорные множества таких рядов являются *рациональными языками*; они соответствуют случаю, когда систему уравнений можно решить с помощью техники «первой степени».

Правые части соответствующих уравнений представляют собой линейные функции неизвестных; коэффициенты расположены с одной стороны от последних.

Пример. *Вычисление некоммутативных рациональных функций.* Рассмотрим систему уравнений

$$A = bB - cC + c, \tag{1}$$

$$B = abB + cA, \tag{2}$$

$$C = bA - cC. \tag{3}$$

Будем обозначать единицу через 1. Уравнение (2) дает

$$(1 - ab)B = cA, \quad \text{откуда} \quad B = \frac{cA}{1 - ab} = (1 + (ab)^*)cA.$$

Аналогично, уравнение (3) дает

$$(1 + c)C = bA, \quad C = \frac{bA}{1 + c} = (1 + (-c)^*)bA.$$

Подставляя полученные значения B и C в (1), имеем

$$A = b\,(1 + (ab)^*)\,cA - c\,(1 + (-c)^*)\,bA + c,$$

$$[1 - b\,(1 + (ab)^*)\,c + c\,(1 + (-c)^*)\,b]\,A = C.$$

Поэтому

$$A = \frac{c}{1 - b\,(1 + (ab)^*)\,c + c\,(1 + (-c)^*)\,b},$$

откуда получаются выражения для B и C.

Данное вычисление оказалось возможным лишь благодаря тому, что входящие в уравнения одночлены имеют специальный вид; все они линейны относительно неизвестных и все их коэффициенты стоят с одной и той же стороны от неизвестных. Это последнее свойство исключает неоднородность размещения множителей и делителей: если в уравнениях системы встречаются двусторонние одночлены или одночлены и типа aA, и типа Aa, то выражения вида

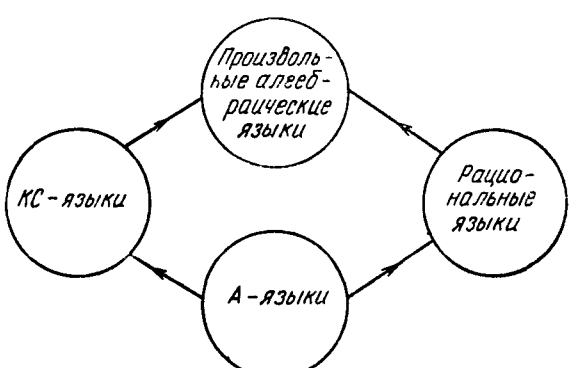
$$A = \frac{b}{1 - a}$$

допускают две различные интерпретации: либо $b(1 + a^*)$, либо $(1 + a^*)b$.

16.3.3. Алгебраические языки

Опорное множество алгебраического ряда является *алгебраическим языком.*

Два признака: «произвольный/алгебраический» и «коэффициенты являются положительными/произвольными целыми», позволяют различать четыре класса языков:



Все включения здесь являются строгими; существуют, например, рациональные языки, не являющиеся А-языками, но являющиеся КС-языками.

16.3.4. Адамаровское произведение рядов

Среди основных операций над языками есть одна, которую мы до сих пор не интерпретировали в терминах рядов, — это операция пересечения. Для такой интерпретации можно использовать понятие адамаровского произведения. Напомним определение адамаровского произведения двух рядов (с коммутативными переменными).

Пусть имеются функции f и g, заданные разложением в целочисленные ряды:

$$f = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n + \ldots,$$
$$g = b_0 + b_1 x + b_2 x^2 + \ldots + b_n x^n + \ldots.$$

Адамаровским произведением $f \odot g$ рядов f и g называется выражение

$$f \odot g = a_0 b_0 + a_1 b_1 x + a_2 b_2 x^2 + \ldots + a_n b_n x^n + \ldots.$$

Для функций одной переменной имеет место следующая

Теорема (Юнген). а) *Если f — рациональная функция и g — рациональная функция, то $f \odot g$ — тоже рациональная функция.*

б) *Если f — рациональная функция и g — алгебраическая функция, то $f \odot g$ — алгебраическая функция.*

в) *Если f — алгебраическая функция и g — алгебраическая функция, то $f \odot g$ может не быть алгебраической функцией.*

Теперь мы обобщим понятие адамаровского произведения следующим образом. Пусть σ_1 и σ_2 — формальные ряды с некоммутативными переменными, и f обозначает произвольную цепочку, принадлежащую опорным множествам обоих рядов σ_1 и σ_2; пусть, кроме того, α и β — степени неоднозначности цепочки f в σ_1 и σ_2 соответственно. Тогда

$$\sigma_1 \odot \sigma_2 = \sum_f \alpha\beta f.$$

Из определения ясно, что опорное множество адамаровского произведения двух рядов совпадает с пересечением опорных множеств этих рядов:

$$\operatorname{supp}(\sigma_1 \odot \sigma_2) = \operatorname{supp}(\sigma_1) \cap \operatorname{supp}(\sigma_2).$$

Теорема Юнгена, обобщенная на некоммутативные функции (ряды), приобретает следующий вид.

Теорема (Юнген — Шютценберже). а) *Если* $\operatorname{supp}(\sigma_1)$ — *рациональный язык и* $\operatorname{supp}(\sigma_2)$ — *рациональный язык, то* $\operatorname{supp}(\sigma_1 \odot \sigma_2)$ — *тоже рациональный язык.*

б) *Если* $\operatorname{supp}(\sigma_1)$ — *рациональный язык и* $\operatorname{supp}(\sigma_2)$ — *алгебраический язык, то* $\operatorname{supp}(\sigma_1 \odot \sigma_2)$ — *алгебраический язык.*

в) *Если* supp (σ_1) — *алгебраический язык и* supp (σ_2) — *алгебраический язык, то* supp $(\sigma_1 \odot \sigma_2)$ *может не быть алгебраическим языком.*

Эта теорема обобщает также результаты, относящиеся к пересечениям КС-языков и А-языков.

Имеет место следующее важное свойство.

Рассмотрим гомоморфное отображение цепочек некоторого языка на язык, задаваемый рядом с коммутативными переменными [1]). При таком отображении указанные выше свойства рядов сохраняются.

Ясно, что это отображение преобразует систему некоммутативных уравнений в классическую систему.

Можно доказать, что верно и обратное.

§ 16.4. УПРАЖНЕНИЯ

1. Функция φ, определенная на кольце $U = \{a, b, c, \ldots\}$ и принимающая действительные значения, называется ***нормой***, если она удовлетворяет следующим условиям:

1) $(\forall a \in U)[\varphi(a) \geqslant 0]$,

2) $\varphi(a) = 0 \Leftrightarrow a = 0$,

3) $\varphi(a + b) \leqslant \varphi(a) + \varphi(b)$,

4) $\varphi(1) = 1$,

5) $\varphi(ab) \leqslant \varphi(a)\varphi(b)$.

Доказать, что $\mathrm{val}[(s)]$ — норма для $\Omega[[X^*]]$.

2. Показать, что выбор числа, превосходящего 1, которое служит для построения функции $\mathrm{val}[(s)]$ (у нас в качестве такого числа была выбрана двойка), не имеет принципиального значения.

3. Доказать утверждение п. 16.1.9, а именно: подстановка рядов (s_j) в произвольный ряд возможна, если ряды (s_j) квазирегулярны.

4. Описать порождение языка

$$\{a^p b^q \mid p, q > 0, p \neq q\}$$

— посредством системы уравнений,

— посредством КС-грамматики.

§ 16.5. ПРИМЕНЕНИЕ «ЯЗЫКОВЫХ» УРАВНЕНИЙ В КОМБИНАТОРНОЙ ГЕОМЕТРИИ

Теория формальных грамматик имеет интересные приложения в некоторых вопросах комбинаторной геометрии и комбинаторного анализа. Мы ограничимся в этой связи разбором примера, пока-

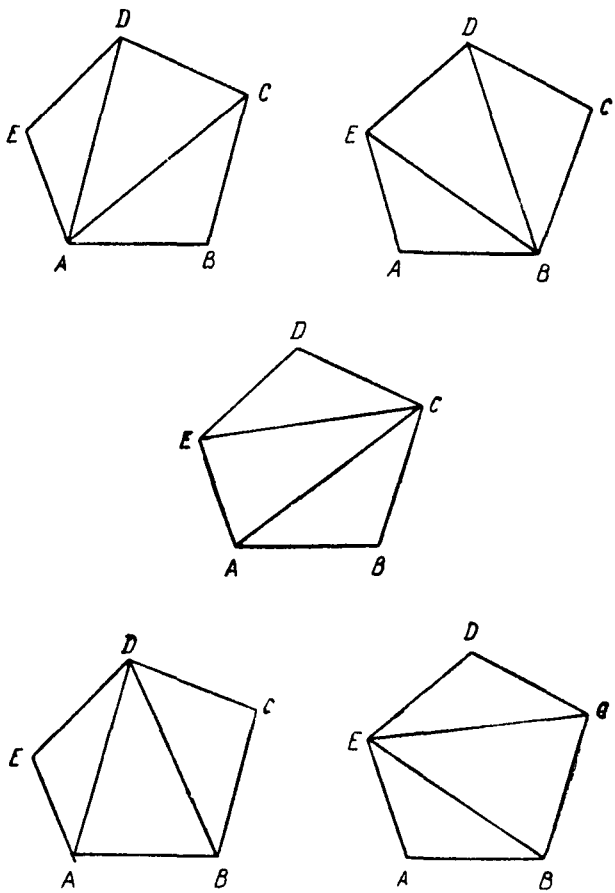
[1]) Точнее, свободной полугруппы на свободную полугруппу с одной образующей. — *Прим. ред.*

зывающего, как с помощью формальных грамматик можно по-новому подойти к решению одной классической задачи.

16.5.1. Задача о разбиении многоугольника

Пусть имеется жорданов многоугольник (т. е. топологический многоугольник, разбивающий плоскость на внешнюю и внутреннюю области) с помеченными вершинами; спрашивается, сколькими разными способами можно разбить его на треугольники посредством диагоналей, не имеющих общих точек, кроме вершин?

Пример. В случае пятиугольника решение очевидно: пятиугольник допускает пять различных разбиений. Вот они:



Наметим сначала путь «классического» решения сформулированной задачи.

Пусть имеется жорданов m-угольник, и пусть $f(m)$ — число его возможных разбиений. Рассмотрим n-угольник P и определим с помощью функции f число разбиений, имеющих один фиксированный общий треугольник. Если выбросить из P этот общий треугольник, останутся два многоугольника, имеющие p и q сторон соответственно, причем

$$p+q=n+1.$$

Следовательно, искомое число есть $f(p)f(q)$.

Если один из двух «малых» многоугольников пуст, данное число равно $f(n-1)$.

Полагая $f(2)=1$, получаем для всех $n=2, 3, \ldots$ единое выражение

$$f(p)f(q); \quad p+q=n+1.$$

Фиксируем одну из сторон n-угольника $A_1A_2\ldots A_n$, например A_1A_2; произвольную вершину, отличную от A_1 и A_2, будем обозначать через A_i.

Все треугольники $A_1A_2A_i$ различны, и всякая триангуляция, содержащая $A_1A_2A_i$, отличается от триангуляции, содержащей $A_1A_2A_j$, $i\neq j$. Суммируя по i, мы получим следующую рекуррентную формулу:

$$f(n)=f(2)f(n-1)+\ldots+f(i)f(n-i+1)+\ldots+f(n-1)f(2).$$

В случае треугольника имеем $f(3)=1$, что позволяет последовательно вычислять $f(n)$ для любых n.

Пример.

$$f(4)=f(2)f(3)+f(3)f(2)=2,$$

$$f(5)=f(2)f(4)+f(3)f(3)+f(4)f(2)=5.$$

Теперь попытаемся определить производящую функцию

$$y=f(2)x^2+f(3)x^3+\ldots+f(n)x^n+\ldots,$$

такую, чтобы коэффициент при x^n давал $f(n)$. Предыдущие рассуждения не подсказывают пути к этому, и создается впечатление, что необходимо прибегнуть к какому-то «трюку». Между тем использование формальных грамматик приводит к весьма естественному решению.

16.5.2. Решение задачи о разбиении многоугольника с помощью грамматики

Основная идея состоит в том, чтобы построить грамматику, порождающую многоугольники, разбитые на треугольники, так, как порождаются цепочки формального языка. Если эта грамматика будет однозначной, то она выдаст — в качестве «побочного продукта» — пересчет треугольников, т. е. число $f(n)$.

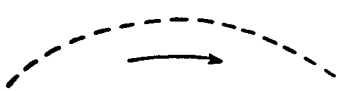
Рассмотрим «бесскобочную» грамматику

$$S \to aSS,$$

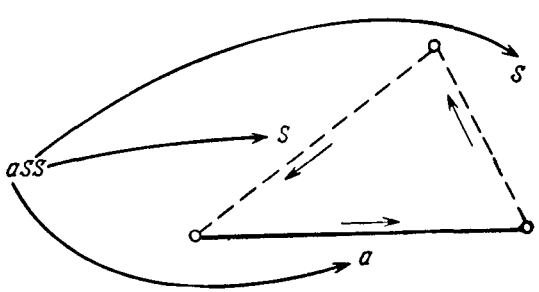
$$S \to b.$$

Эта грамматика однозначна. Дадим ее символам и правилам следующую интерпретацию.

1) S соответствует ориентированному топологическому сегменту, который мы назовем *потенциальным* и будем изображать так:



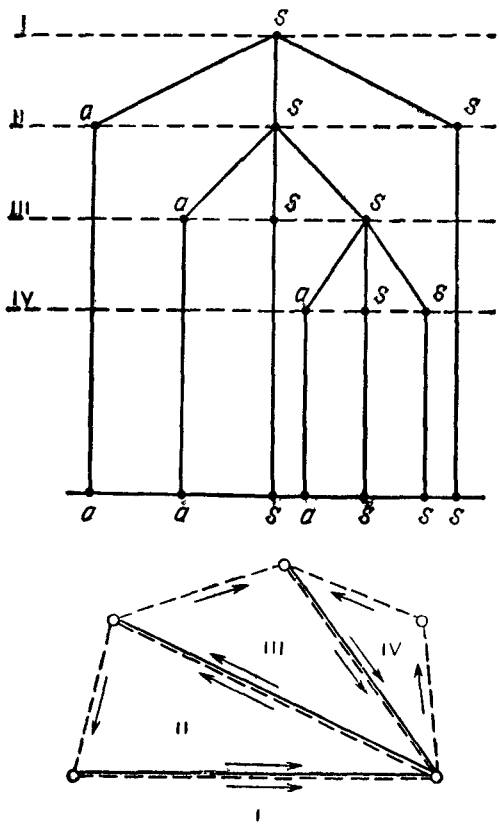
2) aSS отвечает ориентированному топологическому треугольнику, имеющему одну *реальную* сторону a и две *потенциальные* стороны S и S. Это соответствие иллюстрируется следующей схемой:



3) Замена S на aSS (при выполнении первого правила) означает: построить треугольник aSS, такой, что его реальная сторона a совпадает — *с учетом ориентации!* — с соответствующей потенциальной стороной S, а он сам находится вне многоугольника, уже построенного к этому моменту.

Нетрудно доказать по индукции, что это последнее условие всегда может быть соблюдено: тем самым применение правил не связано никакими контекстными ограничениями.

Пример. Приведем дерево вывода цепочки в нашей грамматике и триангулированный многоугольник, являющийся переводом этой цепочки.



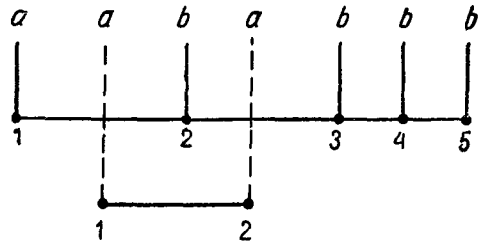
4) Правило $S \to b$ означает замену потенциальной стороны S реальной стороной b. Это заключительное правило замыкает многоугольник с той стороны, где оно применяется.

Ясно, что всякой терминальной цепочке соответствует правильно триангулированный многоугольник. Все a, кроме первого, представляют диагонали, а все b — стороны.

Легко доказать по индукции, что общее число всех S и b равно увеличенному на единицу числу всех a.

С другой стороны, треугольников столько же, сколько имеется вхождений символа a. Таким образом, мы получили классический результат: число треугольников равно числу сторон минус два или числу триангулирующих диагоналей плюс единица.

Пример. Цепочка



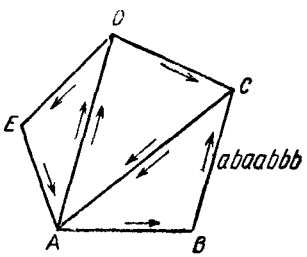
представляет пятиугольник, разбитый двумя диагоналями на три треугольника.

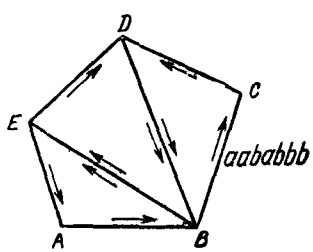
Заметим, что пометить и ориентировать одну сторону многоугольника — это то же самое, что пометить все его вершины. Следовательно, любая терминальная цепочка, содержащая n вхождений a, соответствует *одной* триангуляции некоторого $(n+2)$-угольника с помеченными вершинами.

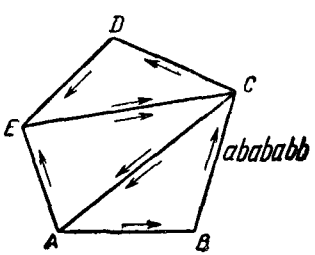
Две разные цепочки, содержащие n вхождений a, отвечают двум разным триангуляциям.

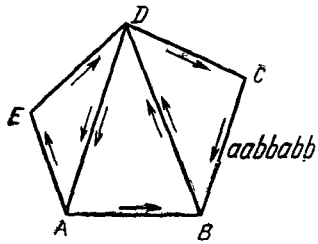
Любая триангуляция может быть получена описанным выше способом.

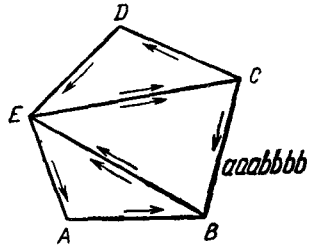
Пример. Взяв пятиугольник $ABCDE$ и выделив в нем сторону AB, мы получим следующие соответствия:











Для того чтобы обеспечить порождение (без повторений) последовательности всех триангулированных многоугольников, достаточно образовать последовательность цепочек, порождаемых грамматикой

$$|\, S \to b, \quad S \to aSS \,|,$$

т. е. решить уравнение, содержащее некоммутирующие члены:

$$S = b + aSS.$$

В результате соответствующих выкладок (приводившихся в гл. XI) получается следующий некоммутативный ряд:

$$(s):\ b + abb + ababb + aabbb + aabbabb + \ldots .$$

Если нас интересует только подсчет числа возможных разбиений многоугольника, то, поскольку число сторон многоугольника равно числу вхождений в цепочку символа a, увеличенному на два, ясно, что число возможных триангуляций $(n+2)$-угольника равно коэффициенту при x^n в коммутативном ряде, полученном из (s) заменой a на x и b на 1.

Достаточно рассмотреть гомоморфизм

$$\begin{cases} a \to x, \\ S \to y, \quad xy = yx, \\ b \to 1. \end{cases}$$

Функция y удовлетворяет уравнению

$$y = 1 + xy^2.$$

Имеем

$$y = 1 + x + 2x^2 + 5x^3 + \ldots .$$

Пример. Для пятиугольника $(n + 2 = 5)$ надо взять коэффициент при x^3, т. е. 5.

16.5.3. Упражнения

1. Пусть имеется треугольник ABC. Взяв внутри него некоторую точку и соединив ее с вершинами, мы разобьем треугольник

на три других, каждый из которых может аналогичным образом быть разбит на три треугольника и т. д.

Исследовать эту *тернарную триангуляцию*. Использовать грамматику

$$\left|\begin{array}{l} T \to TTT \\ T \to t \end{array}\right|.$$

2. Исследовать разбиение $2n$-угольника на четырехугольники с помощью непересекающихся диагоналей.

Приложение

ТРАНСФОРМАЦИОННЫЕ ГРАММАТИКИ

§ 1. ФОРМАЛЬНЫЕ ГРАММАТИКИ И ЕСТЕСТВЕННЫЕ ЯЗЫКИ

Многие понятия, рассматривавшиеся в этой книге, возникли при исследовании естественных языков [1]). Например:

— В логике были формализованы определенные операции над предложениями обычного языка: «неверно, что ...»: $\neg$, «... или... ...»: $\vee$, «... и ...»: $\wedge$, «если ..., то...»: $\supset$, «... равносильно...»: $\equiv$ и т. п. (вместо многоточий должны подставляться предложения).

— Возникновение понятия марковской цепи, соответствующего понятию А-языка, было связано со статистическим исследованием последовательностей гласных и согласных в литературном тексте.

— Понятие формальной грамматики, введенное Н. Хомским, возникло в результате многочисленных усилий, направленных на создание строгих и абсолютно эксплицитных описаний некоторых грамматических закономерностей естественного языка (Хэррис [1946—1951]).

Цель современной лингвистики состоит в построении таких описаний естественных языков, которые могли бы быть полностью формализованы с помощью абстрактного автомата, формальной грамматики или какой-либо алгебраической конструкции. Такое требование к лингвистическим описаниям представляется совершенно обоснованным с теоретической точки зрения: ведь лингвист хочет познать и описать не что иное, как механизмы, посредством которых люди строят и понимают высказывания. Эти механизмы, образующие языковую интуицию носителей данного языка, описываются традиционной грамматикой весьма несовершенно.

Формализация лингвистических описаний должна повести к выявлению новых закономерностей и к формулировке весьма общих правил, допускающих как можно меньше исключений. Благодаря формализации станет возможным сравнение грамматик самых разных по строению языков и выявление стоящих за ними универсальных механизмов, допускающих математическое исследование.

С практической точки зрения создание достаточно полных формализованных грамматик необходимо для различных направлений той научной области, которая известна под названием «автоматическая обработка информации», например для автоматического

---

[1]) Правильнее было бы сказать: «так или иначе восходят к естественному языку». — *Прим. ред.*

анализа и синтеза текстов с помощью электронных вычислительных машин.

С начала 50-х годов исследования, направленные на создание формализованных лингвистических описаний, которые опирались бы на богатый фактический материал, велись весьма интенсивно. В ходе этих исследований выяснилось, в частности, что в плане описания синтаксических явлений естественного языка комбинаторные системы, рассмотренные в настоящей книге, обладают рядом недостатков, объясняющихся, прежде всего, недостаточной специализированностью названных систем. Это привело к созданию еще одного способа описания синтаксиса, известного под именем «трансформационного анализа» (см. Хэррис [1952], [1957], [1964]; Хомский [1957], [1962], [1965]).

Основная идея трансформационного анализа состоит в следующем. Всякая фраза (предложение) естественного языка может быть описана как состоящая из некоторого числа простых фраз (предложений), называемых *ядерными* (англ. kernel sentences), из которых данная фраза образуется посредством определенных операций — так называемых *трансформаций*. Ядерные фразы распадаются в свою очередь на еще более простые элементы — слова (= словоформы) или, быть может, корни и аффиксы. (В действительности простейшими единицами, или *атомами*, подобного описания являются не реальные слова или корни и аффиксы, а некоторые более абстрактные сущности, определяемые так, чтобы обеспечить максимальную компактность и единообразие описаний при минимальном числе этих атомов.)

Необходимо подчеркнуть, что указанный подход представляет собой вполне традиционный научный прием описания явлений природы. Любопытно, что близкий к этому метод широко применялся в так называемых рациональных, или логических, грамматиках 17 и 18 веков, но позже был отвергнут и практически изгнан из лингвистики на двести лет, несмотря на то, что выдвигавшиеся против него аргументы носили крайне сомнительный характер.

Вот пример трансформации, четко изложенной в знаменитой «Грамматике Пор-Рояля» А. Арно и К. Лансело (1660 г.). Фраза

(1) *Dieu invisible a créé le monde visible* 'Невидимый бог создал видимый мир',

анализируется следующим образом:

(2) *Dieu qui est invisible a créé le monde qui est visible* 'Бог, который является невидимым, создал мир, который является видимым'.

(3) (i) *Dieu a créé le monde* 'Бог создал мир'.
(ii) *Dieu est invisible* 'Бог является невидимым'.
(iii) *Le monde est visible* 'Мир является видимым'.

Таким образом,

(3) — это совокупность трех ядерных фраз;

(2) состоит из основной ядерной фразы (англ. matrix sentence) (3i), в которую вставлены фразы (3ii) и (3iii). Операция, посредством которой выполнено это вставление, есть трансформация;

(1) получена из (2) особой трансформацией элиминации (*qui est invisible* $\Rightarrow$ *invisible*).

Преимущество такого описания становится очевидным, если сравнить его, например, с описанием (повсеместно принятым в лингвистике) в терминах КС-грамматики.

§ 2. КС-ГРАММАТИКИ И ТРАНСФОРМАЦИИ

Рассмотрим следующее множество правил, записанных в виде уравнений (по существу это КС-грамматика):

$$(A)^1) \quad : \quad \left| \begin{array}{l} Ph = Gn \;\; Gv \\ Gn = Art \;\; N_{\text{нар}} + N_{\text{соб}} \\ Gv = V \;\; Gn + est \;\; Adj \\ Art = le \\ V = a \;\; créé \\ N_{\text{нар}} = monde \\ N_{\text{соб}} = Dieu \\ Adj = visible + invisible \end{array} \right\} \text{заключительные, или словарные, правила}$$

$N_{\text{нар}}$ обозначает нарицательное существительное, $N_{\text{соб}}$ — имя собственное; значения остальных символов пояснены на стр. 114.

КС-грамматика (A) порождает 8 фраз, которые можно представить формулой

(*Dieu* + *le monde*) (*a créé* (*Dieu* + *le monde*) + *est* (*visible* + + *invisible*)).

NB: здесь скобки использованы, разумеется, не для того, чтобы показать структуру, а просто для записи произведения.

Чтобы получить фразы (1) и (2), необходимо добавить к (A) правила, вводящие определения к существительным:

$$(B) \quad : \quad \left| \begin{array}{ll} (B1) & Gn = Art \;\; N_{\text{нар}} \;\; Adj + N_{\text{соб}} \;\; Adj \\ & \text{(определения-прилагательные)} \\ (B2) & Gn = Art \;\; N_{\text{нар}} \;\; qui \;\; Gv + N_{\text{соб}} \;\; qui \;\; Gv \\ & \text{(определительные придаточные)} \end{array} \right.$$

В грамматике (A, B) имеются циклические правила, вследствие чего она порождает бесконечное множество фраз, содержащих определительные придаточные, последовательно вставляемые одно в другое (*qui Gv* в правой части правила ($B2$)).

---

[1]) Знак + здесь и ниже означает разделительное «или». — *Прим. перев.*

Такое описание предполагает иное содержательное представление об определениях существительного, чем то, которое диктуется трансформационным подходом. А именно, здесь правила (B) непосредственно вводят определения во фразу, и притом независимым образом для каждого из двух типов определения $(B1)$ и $(B2)$. Напротив, при трансформационном подходе подчеркивается одинаковый (с точностью до правила элиминации) статус прилагательного в (1) и (2).

Рассмотрим теперь трансформационную грамматику, порождающую язык $L(A, B)$; она состоит из двух компонентов:

. КС-грамматика (A), порождающая ядерные фразы;

.. множество (C) трансформаций, определенных на структурах таких фраз, которые либо являются ядерными, либо получаются применением трансформаций.

Множество (C) распадается на два подмножества.

— Бинарные трансформации (определенные на парах фраз). Примером может служить вставление определительных придаточных предложений в главное предложение; эту операцию можно записать в виде

$$(C1) \; : \; X \; N_1 \; Y, \; Art \; N_2 \; Gv \rightarrow X \; N \; qui \; Gv \; Y,$$

где $N_1 = N_2 = N$; X и Y — левый и правый контексты существительного N_1 в главном предложении.

— Унарные трансформации (определенные на фразах), например элиминация выражения *qui est*:

$$(C2) \; : \; X \; N \; qui \; est \; Adj \; Y \rightarrow X \; N \; Adj \; Y.$$

Предлагаемый подход может, однако, вызвать следующее возражение. Введенные нами трансформационные правила очень сложны и поэтому их трудно изучать; при этом они не задают никаких операций, которые не описывались бы правилами (B); кроме того, грамматика (A, B) гораздо более однородна по своему строению, чем грамматика (A, C).

§ 3. РАСШИРЕНИЕ ГРАММАТИКИ

Только что приведенное возражение представляется убедительным лишь на первый взгляд. В самом деле, допустим, что мы хотим существенно расширить наше описание и охватить гораздо больше разных языковых явлений, чем их имеется в $L(A, B)$. Пусть, например, нам нужно учесть наличие разных типов согласования в роде, числе и лице или какие-либо ограничения на семантические классы подлежащих и сказуемых. Для этого потребуется определенным образом изменить обе грамматики — как (A, B), так и (A, C). Оказывается, однако, что внесение необходимых изменений в КС-грамматику ведет к существенному усложнению последней, тогда как для трансформационной грамматики

соответствующая модификация производится естественным и элегантным образом.

Так, чтобы учесть согласование только в роде и числе, в грамматике (A) придется сделать изменения для пар «артикль — существительное», «подлежащее — сказуемое», «существительное — прилагательное в роли определения», «существительное — прилагательное в роли именной части сказуемого определительного придаточного предложения». В результате грамматика (A) превратится в (A') с правилами вида

$$(A'1) \quad : \quad Ph = Gn_{\text{ед}}\ Gv_{\text{ед}} + Gn_{\text{мн}}\ Gv_{\text{мн}}$$

и т. д. Приведенное правило порождает фразы без глагола *être* 'быть'. Для фраз с этим глаголом необходимо согласование не только в числе, но и в роде, для чего потребуется ввести правила вида

$$\begin{aligned}(A'2) : Ph = {} & Gn_{\text{муж, ед}}\ est\ Adj_{\text{муж, ед}} + \\ & + Gn_{\text{жен, ед}}\ est\ Adj_{\text{жен, ед}} + \\ & + Gn_{\text{муж, мн}}\ sont\ Adj_{\text{муж, мн}} + \\ & + Gn_{\text{жен, мн}}\ sont\ Adj_{\text{жен, мн}} \text{ и т. д.}\end{aligned}$$

Фигурирующие в ($A'1$) и ($A'2$) именные группы имеют совершенно одинаковую природу. Однако для того, чтобы показать это в явной форме, нам пришлось бы продублировать правило ($A'1$) из-за наличия Gn разного рода, что с лингвистической точки зрения нежелательно: во французском языке род никогда не участвует в согласовании сказуемого-глагола с подлежащим. Кроме того, поскольку род и число — независимые параметры, понадобятся еще и правила:

$$\begin{aligned}(A'3) \quad : \quad & Gn_{\text{ед}} = Gn_{\text{муж, ед}} + Gn_{\text{жен, ед}} \\ & Gn_{\text{мн}} = Gn_{\text{муж, мн}} + Gn_{\text{жен, мн}}.\ ^{1})\end{aligned}$$

(Введение глагольных групп с индексами рода представляло бы собой еще более громоздкое решение.)

Далее, необходимы также следующие правила:

$$\begin{aligned}Gn_{\text{муж, ед}} &= le\ N_{\text{нар, муж, ед}} + N_{\text{соб, муж, ед}} \\ Gn_{\text{муж, мн}} &= les\ N_{\text{нар, муж, мн}} + N_{\text{соб, муж, мн}} \\ Gn_{\text{жен, ед}} &= la\ N_{\text{нар, жен, ед}} + N_{\text{соб, жен, ед}} \\ Gn_{\text{жен, мн}} &= les\ N_{\text{нар, жен, мн}} + N_{\text{соб, жен, мн}} \\ Gv_{\text{ед}} &= V_{\text{ед}}\ (Gn_{\text{ед}} + Gn_{\text{мн}}) \\ Gv_{\text{мн}} &= V_{\text{мн}}\ (Gn_{\text{ед}} + Gn_{\text{мн}}).\end{aligned}$$

[1]) Это правило необходимо ввиду наличия Gn в правой части одного из правил грамматики (A), имеющих в левой части Gv. — *Прим. перев*

Наконец, нужны заключительные (словарные) правила:

$$N_{\text{нар, муж, ед}} = monde + \ldots$$
$$N_{\text{нар, муж, мн}} = mondes + \ldots$$
$$N_{\text{соб, муж, ед}} = Dieu + \ldots$$
$$N_{\text{соб, муж, мн}} = Dieux + \ldots$$
$$N_{\text{нар, жен, ед}} = table + \ldots$$
$$N_{\text{нар, жен, мн}} = tables + \ldots$$
$$N_{\text{соб, жен, ед}} = Marie + \ldots$$
$$N_{\text{соб, жен, мн}} = Maries + \ldots$$
$$Adj_{\text{муж, ед}} = visible + invisible + beau + loyal + \ldots$$
$$Adj_{\text{жен, ед}} = visible + invisible + belle + loyale + \ldots$$
$$Adj_{\text{муж, мн}} = visibles + invisibles + beaux + loyaux \ldots$$
$$Adj_{\text{жен, мн}} = A\,dj_{\text{жен, ед}}\,s$$

(Эти правила не являются, разумеется, исчерпывающими: они не учитывают многочисленных морфологических «неправильностей».)

Правила (B) превращаются в правила (B'):

$$(B')\ :\quad Gn_{\text{муж, ед}} = le\ N_{\text{нар, муж, ед}}\ Adj_{\text{муж. ед}} + N_{\text{соб, муж, ед}}\ Adj_{\text{муж, ед}}$$
$$Gn_{\text{муж, ед}} = (le\ N_{\text{нар, муж, ед}} + N_{\text{соб, муж, ед}})\ qui\,(Gv_{\text{муж, ед}} + est\ Adj_{\text{муж, ед}})$$

и еще шесть аналогичных уравнений, соответствующих прочим комбинациям рода и числа.

Легко видеть, что введение подклассов «одушевленное/неодушевленное», «конкретное/абстрактное» и т. п. приведет к еще более значительному возрастанию числа правил типов (A') и (B').

Что же касается трансформационных правил типа (C), то здесь имеется возможность избежать существенного увеличения числа правил. А именно, поскольку на эти правила не наложены заранее никакие формальные ограничения, мы можем видоизменить их весьма естественным способом и притом так, что они будут описывать тот же язык $L(A, B)$, но их число не возрастет.

Превратим (A) в (A''), действуя следующим образом.

Будем приписывать вспомогательным символам индексы, на которые можно ссылаться в правилах. Например, всякому существительному N мы припишем три индекса — переменные, отвечающие трем противопоставлениям: нарицательное/собственное, мужской/женский (род), единственное/множественное (число). В правиле может фигурировать либо N без индексов (или, что то же самое, с «незаполненными» индексами: N_{xxx}), либо с одним лишь

вторым индексом (например, $N_{x,\text{муж},x}$), либо с первым и третьим индексами ($N_{\text{нар},x,\text{мн}}$) и т. п. Реальное существительное может быть выбрано из словаря только при условии, что все индексы символа N фиксированы.

В результате мы получим следующие правила:

$Ph = Gn\ Gv$

$Gn = Art_{xx}\ N_{xxx}$ (переменные индексы при Art и два последних переменных индекса при N — это род и число; первый переменный индекс при N — тип, т. е. нарицательное или собственное),

$Gv = V_{xx}Gn + V_{xx,\text{связка}}\ Adj_{xx}$ (переменные индексы при V — лицо и число, «связка» — признак глаголов-связок; переменные индексы при Adj — род и число).

Значения переменных индексов задаются правилами:

$$тип = \text{нар} + \text{соб}$$

$$род = \text{муж} + \text{жен}$$

$$число = \text{ед} + \text{мн}$$

$$лицо = 1 + 2 + 3$$

Это дает контекстно-зависимые правила:

$Art_{xx}\ N_{\text{соб},x,\text{ед}} = E\ N_{\text{соб},x,\text{ед}}$ [1]) (перед именем собственным в ед. числе артикль не употребляется; впрочем, в действительности закономерности употребления артикля перед именами собственными являются гораздо более сложными),

$Art_{xx}\ N_{x,\text{род,число}} = Art_{\text{род,число}}\ N_{x,\text{род,число}}$ (артикль получает род и число того существительного, перед которым стоит).

Как мы видим, это правило не зависит от типа существительного: оно допускает постановку артикля и перед именами собственными. Получение правильных выражений зависит от достаточно тонкой классификации имен собственных.

$$N_{xx,\text{число}}\ V_{xx} = N_{xx,\text{число}}\ V_{3,\text{число}}$$

(согласование сказуемого с подлежащим)

$$N_{x,\text{род,число}}\ V_{xx,\text{связка}}\ Adj_{xx} = N_{x,\text{род,число}}\ V_{xx,\text{связка}}\ Adj_{\text{род,число}}$$

(если сказуемое есть глагол-связка, то прилагательное-предикатив согласуется с подлежащим).

---

[1]) E — пустая цепочка. — *Прим. перев.*

Кроме того, необходимы заключительные (словарные) правила

$$V_{3,\text{ед, связка}} = est + devient + \ldots$$
$$V_{3,\text{мн, связка}} = sont + deviennent + \ldots$$
$$Art_{\text{муж, ед}} = le$$
$$Art_{\text{жен, ед}} = la$$
$$Art_{x,\text{мн}} = les$$
$$N_{\text{соб, муж, ед}} = Dieu$$

и другие словарные правила, указанные выше, при описании A'.

Выписанные здесь контекстно-зависимые правила не подходят под определение НС-правил из п. 12.1.2. В самом деле, они не описывают подстановок в цепочках символов, а задают значения переменных индексов при символах, используя информацию, содержащуюся в индексах при одном из них (при N). Благодаря этому правило ($C1$) сохраняется в неизмененном виде, — если условиться, что символ N в ($C1$) понимается как N с какими угодно индексами. Видоизменив ($C2$) следующим образом:

$$(C'2) \quad : \quad XN\ qui\ V_{xx,\text{связка}}\ Adj\ V \rightarrow XN\ Adj\ Y,$$

мы также покроем все нужные случаи

Введение переменных индексов позволяет упростить трансформационную грамматику по крайней мере в силу двух следующих факторов:

— мы используем контекстно-зависимые правила с переменными индексами, а это выводит собственно порождающий компонент грамматики за рамки класса КС-грамматик;

— трансформации могут применяться независимо от тех или иных индексов. Это обеспечивает возможность существенного обобщения трансформационных правил и одновременно позволяет выразить в явной форме тот факт, что тип, род, число и лицо являются независимыми параметрами (с помощью КС-правил выразить этот факт невозможно).

В КС-грамматиках всякая более тонкая классификация влечет за собой увеличение числа правил; каждый раз, когда в описание вводится новый параметр, требуется пересмотреть всю КС-грамматику. Если же мы пользуемся трансформационной грамматикой, то подобные усовершенствования предполагают изменение только контекстно-зависимых правил компонента (A'') и не затрагивают трансформационных правил [1]).

[1]) Развиваемая здесь аргументация «против КС-грамматики» представляется малоубедительной, поскольку «расщепление» правил, вызванное необходимостью обеспечить согласование, вряд ли вообще следует рассматривать как недостаток. Для сокращенной записи таких «расщепленных» правил можно воспользоваться схемами правил (аналогичными используемым в математической логике схемам

Единственным бесспорным достоинством КС-грамматик с точки зрения их применения для описания естественных языков является удобство их применения для автоматического синтаксического анализа [1]): алгоритмы анализа текста на основе КС-грамматики и соответственно машинные программы их реализации достаточно просты [2]). Однако присущие КС-грамматикам сильные ограничения на форму правил приводят к тому, что описание синтаксической структуры фраз естественного языка в терминах «КС-деревьев», т. е. деревьев составляющих, в ряде случаев оказывается неудовлетворительным в теоретическом отношении. Что же касается практического аспекта, то для признания деревьев составляющих необходимо как минимум наличие эффективной процедуры, которая сопоставляла бы правильное дерево составляющих любой фразе данного естественного языка; однако до такой процедуры еще весьма далеко. Более того, даже если бы эта процедура уже имелась в нашем распоряжении, во многих случаях результаты синтаксического анализа были бы непригодны для каких бы то ни было разумных целей: существует много таких необходимых синтаксических сведений, которые очень трудно или невозможно выразить на языке деревьев составляющих.

Приведем пример подобной трудности. Пусть имеется фраза
(1) *Jean obtient de Pierre l'autorisation de partir* 'Жан полу́чает у Пьера разрешение уехать'.
С помощью дерева составляющих естественным образом выра-

аксиом), содержащими переменные для рода, числа и т. п., причем, разумеется, переменные для разных категорий должны быть разными; например, четыре правила (*A*'3) можно записать в виде схемы $Gn_y \rightarrow Gn_{xy}$, где x, y — переменные, пробегающие соответственно множества {муж, жен} и {ед, мн}. В теоретическом аспекте увеличение числа правил вообще не имеет значения, важна лишь их обозримость и естественность, а с этой точки зрения записанные в виде схем КС-правила ничуть не уступают контекстно-зависимым правилам предлагаемого авторами типа (и даже, можно полагать, превосходят последние). Для практических целей (составление машинных программ и т. п.) схемы КС-правил также не представляют никаких неудобств в сравнении с контекстно-зависимыми правилами.

Сказанное, разумеется, не затрагивает «традиционного» обоснования необходимости трансформаций, исходящего из невозможности объяснить с помощью одних только КС- (или НС-)правил связь между «родственными» предложениями (активное/пассивное; повествовательное/вопросительное и т. п.) — *Прим. ред.*

[1]) С этим утверждением трудно согласиться. КС-грамматики, конечно, не являются полностью адекватным средством описания синтаксиса естественного языка, но многие его аспекты описываются ими достаточно хорошо. Следует также подчеркнуть, что трансформационные грамматики, как правило, содержат существенные компоненты, являющиеся КС-грамматиками. — *Прим. ред.*

[2]) Об алгоритмах анализа КС-языка см, например, Янгер [1967] — *Прим. ред.*

жаются следующие сведения о синтаксическом строении этой фразы:

Jean — подлежащее при сказуемом *obtient* 'получает';

Pierre — предложное дополнение к *obtient*, а *l'autorisation de partir* 'разрешение уехать' — прямое дополнение к тому же глаголу;

partir 'уехать' — дополнение к *l'autorisation* 'разрешение'.

Однако между словами рассматриваемой фразы имеются и другие отношения, выразить которые в терминах составляющих гораздо сложнее, например:

Jean — субъект действия, обозначенного глаголом *partir* 'уехать' (это доказывается тем, что две следующие фразы:

J'obtiens de Pierre l'autorisation de m'en aller 'Я получаю у Пьера разрешение уйти' и *Il obtient de Pierre l'autorisation de s'en aller* 'Он получает у Пьера разрешение уйти', — правильны, а фраза

* *Il obtient de Pierre l'autorisation de m'en aller* 'Он получает у Пьера разрешение, чтобы я ушел' — неправильна);

Pierre — субъект действия, обозначенного группой *l'autorisation de partir*, т. е. Пьер дает разрешение (это следует из того, что фраза *Jean a vortre autorisation de partir* 'Жан имеет ваше разрешение уехать' — правильна, а фраза **Jean obtient de Pierre votre autorisation de partir* 'Жан получает у Пьера ваше разрешение уехать' — неправильна).

Оба указанных отношения невозможно естественным образом выразить в терминах КС-грамматики. Действительно, естественный способ описать эти отношения таков: сказать, что фраза (1) «содержит в себе» фразу (2):

(2) *Pierre autorise Jean à partir* 'Пьер разрешает Жану уехать', и что (1) получается путем определенной перестройки (=«номинализации») фразы (2).

Трансформационные правила позволяют легко описывать операции подобного типа.

§ 4. ПРОБЛЕМЫ, СВЯЗАННЫЕ С ТРАНСФОРМАЦИЯМИ

Основная идея трансформационного подхода к лингвистическому описанию фраз естественного языка состоит в использовании «атомов» (т. е. элементарных символов) двух уровней. Атомы первого уровня — это морфемы (корни, окончания). Из них посредством НС-правил (в частности, КС-правил) строятся так называемые ядерные предложения (kernel sentences). Эти ядерные предложения выступают как атомы второго уровня; из них посредством трансформационных правил строятся более сложные предложения. Ядерные предложения имеют ограниченную длину; однако,

поскольку трансформации определяются рекурсивно и могут соединять любые, в том числе и неядерные предложения, длина фраз, которые могут быть получены в результате применения трансформаций, в принципе не ограничена.

Кроме того, мы позволили себе подвергнуть грамматику (A, C) достаточно глубоким изменениям. В то же время в рамках грамматики (A, B) круг допустимых изменений жестко ограничен требованием использовать только КС-правила. Свобода выбора типа правил — это по существу основная проблема при построении грамматики.

Однако из лингвистических исследований, посвященных изучению синтаксиса различных естественных языков, видно, что чем больше сведений мы хотим включить в трансформационные правила, тем меньше у нас свободы в выборе формы этих правил. Кроме того, стремление придать правилам простой и вместе с тем логически отчетливый вид приводит к целому ряду добавочных формальных ограничений.

Изложим теперь вкратце существующие представления о трансформационной грамматике.

(1) «Исходная» концепция З. Хэрриса, развитая впоследствии Н. Хомским [1957]. Сущность этой концепции может быть проиллюстрирована правилами (C) из нашего примера, в которых, однако, род и число должны теперь трактоваться как отдельные морфемы.

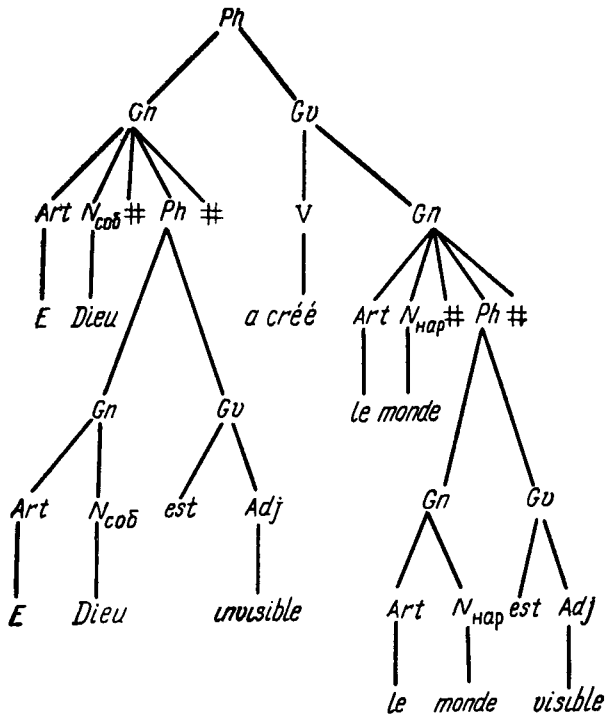
(2) «Новая» концепция Н. Хомского [1965] предполагает порождающую грамматику, устроенную следующим образом. Базовый компонент этой грамматики представляет собой систему КС-правил, порождающую глубинно-синтаксические структуры. Затем эти структуры конкретизируются: с помощью контекстно-зависимых правил переменным индексам придаются определенные значения (так, как это было показано на стр. 278). Далее, глубинно-синтаксические структуры преобразуются — посредством трансформационных правил — в поверхностно-синтаксические структуры. Указанное выше различие между унарными и бинарными трансформациями здесь теряет смысл, поскольку в глубинной структуре представлены все элементарные (т. е. ядерные) предложения, из которых строится сложное предложение.

Грамматика такого типа, необходимая для описания языка $L(A, B)$, должна состоять из грамматики (A'') и правила, которое обеспечивало бы постановку предложений, являющихся определениями к существительным, в позицию после соответствующих существительных:

$$Gn = Art_{xx}\ N_{xxx}\ \#\ Ph\ \#.$$

(# — граничный символ, обозначающий начало и конец элементарного предложения).

Глубинная структура (ГС), отвечающая фразам (1) и (2), такова:



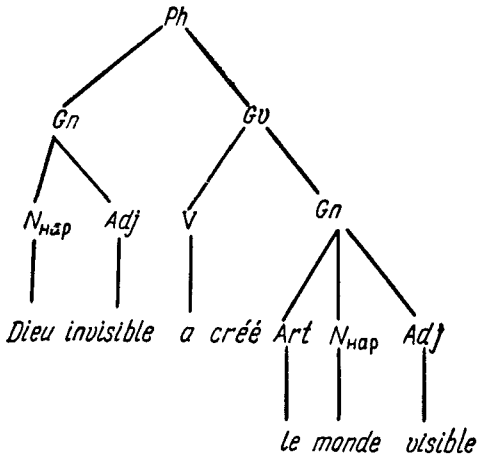
На этом уровне возникает, в частности, следующая проблема: лексемы (= терминальные символы, которыми помечены узлы структуры) должны быть заданы так, чтобы подлежащие тех предложений, которые станут придаточными относительными, были идентичны антецедентам в соответствующих главных предложениях. Обеспечить решение указанной проблемы можно разными способами (ср. Хомский [1965] — Клайма [1965]).

Переход от ГС к поверхностной структуре (ПС1) фразы (1) выполняется с помощью двух следующих трансформационных правил:

— Первое правило применяется к ГС; оно стирает граничные символы # и замещает все «дублирующие» *Gn* относительным местоимением *qui* 'который'. В результате получается ПС2 — поверхностная структура фразы (2); эту структуру мы здесь выписывать не будем.

— Второе правило, а именно (*C*2), применяется к ПС2. Оно стирает *qui est* и должным образом перестраивает дерево, так что

в итоге мы получаем ПС1:



Следует отметить, что многие проблемы, связанные с трансформациями, в том числе с самим понятием трансформации, еще далеки от окончательного решения. Это становится особенно ясно, если попытаться определить трансформации как функции, определенные на множестве КС-деревьев и принимающие значения в множестве деревьев. На современном этапе разработки теории трансформационных грамматик понятие трансформации представляется слишком сложным для строгого математического анализа и описания [1]).

(3) «Новая» концепция З. Хэрриса [1964] более алгебраична по своему характеру. Она предполагает базовое множество образующих — морфем, соответствующих разным частям речи, и конечный набор операций, посредством которых из образующих строятся ядерные предложения. Эти последние составляют «множество образующих второго порядка»; из них посредством унарных и бинарных операций строятся остальные предложения.

Рассмотрим ядерные предложения (3i — 3iii), стр. 273, и следующие трансформации:

(*C*1) $T_r(P_1, P_2)$: P_2 вставляется в P_1 в качестве придаточного относительного;

(*C*2) $T_a(P)$: в P стирается *qui est*.

С их помощью можно получить ряд сложных предложений:
T_r(3i, 3ii) = *Dieu qui est invisible a créé le monde*;

[1]) Ср., однако, работу Гинзбурга и Парти [1969], где предложен математический аппарат для описания трансформаций в смысле Н. Хомского. — *Прим. ред.*

T_r(3i, 3iii) = *Dieu a créé le monde qui est visible*;

$T_r(T_r$(3i, 3ii), 3iii) = $T_r(T_r$(3i, 3iii), 3ii) = *Dieu qui est invisible a créé le monde qui est visible*;

$T_a(T_a(T_r(T_r$(3i, 3ii), 3iii))) = $T_a(T_a(T_r(T_r$(3i, 3iii), 3ii))) = *Dieu invisible a créé le monde visible.*

Заметим, что это последнее предложение может быть порождено и другим способом, а именно:

$$T_a(T_r(T_a(T_r(3i,\ 3ii)),\ 3iii)) = T_a(T_r(T_a(T_r(3i,\ 3iii)),\ 3ii)).$$

Из этих примеров видно, с какого типа проблемами приходится сталкиваться при построении алгебры трансформаций. Так, фраза (1) допускает четыре разных синтаксических описания; однако было бы желательно сопоставлять фразе разные синтаксические описания только в том случае, если она неоднозначна, и притом так, чтобы различные описания соответствовали ее различным смыслам. Поэтому в случае фразы (1) либо все ее четыре синтаксических описания должны считаться эквивалентными, либо сам процесс порождения должен быть организован так, чтобы он давал для однозначной фразы единственное описание.

Основываясь на эмпирических соображениях, мы можем достаточно естественным образом разбить совокупность трансформаций на классы. Например, некоторые унарные трансформации (типа пассивизации или «рефлективизации») образуют особый класс: они должны применяться до всех бинарных трансформаций. Другой класс образуют сугубо локальные (морфонологические) унарные трансформации: они применяются в самом конце процесса порождения. Если отнести трансформацию T_a к этому последнему типу, то все T_a должны помещаться в начало любого произведения трансформаций[1]). Это будет возможно, если постулировать перестановочность трансформаций T_a и T_r.

Возникает также проблема ассоциативности произведения трансформаций; так, умножение трансформаций T_r ассоциативно, тогда как для многих других трансформаций умножение не ассоциативно. Хэррис рассматривает и иные формальные свойства трансформаций: автоматность цепочек трансформаций, определение обратной трансформации (для некоторых частных случаев) и т. п.

Заметим, что все проблемы, связанные с уточнением понятия трансформации в смысле Хомского, возникают и для трансформаций Хэрриса. С другой стороны, проблема выделения допустимых последовательностей трансформаций Хэрриса (пока что это делается эмпирически) сохраняет свое значение и при подходе Хомского.

---

[1]) Произведением двух трансформаций **естественно** назвать трансформацию, получаемую в результате последовательного **выполнения** исходных трансформаций (в заданном порядке). — *Прим. ред.*

Бегло описав три названные выше концепции, мы попытались дать представление об общей проблеме построения формализованного синтаксиса для естественных языков. Эта проблема является полностью эмпирической. Именно исследования многочисленных синтаксических явлений естественных языков (направленные, в частности, на отличение допустимых цепочек словоформ от недопустимых) легли в основу разработки синтаксических формализмов. В отдельных случаях эти формализмы удается абстрагировать от исходных данных и изучать независимо; однако лингвисту должно быть ясно, что все предлагавшиеся формализмы весьма далеки от совершенства и что для построения исчерпывающего формального описания синтаксиса понадобятся еще многочисленные и трудоемкие исследования содержательного характера. Так, до сих пор неизвестно, в какой степени отвечают такие формализмы, как НС- и КС-грамматики, языковой или психологической реальности. В самом деле, в ходе исследований встречаются ситуации наподобие следующей:

— на некотором этапе исследования определенная группа явлений (α) хорошо описывается контекстно-зависимыми правилами ($\mathfrak{G}$):

— на более позднем этапе лингвист встречается с явлениями (β), для описания которых требуются трансформационные правила ($\mathfrak{T}$);

— оказывается, что правила ($\mathfrak{T}$) позволяют описать более естественным образом и явления (α). Лингвист отказывается от правил ($\mathfrak{G}$) и принимает единообразную трактовку (α) и (β) в терминах правил ($\mathfrak{T}$).

При таких условиях — а это условия, вполне нормальные для молодой и развивающейся науки, — сейчас трудно сказать, какой характер будут носить в дальнейшем абстрактные исследования формальных моделей естественных языков. Ясно, однако, что следует всячески приветствовать попытки, направленные на систематическое повышение содержательной адекватности абстрактных моделей языка: эти попытки предполагают параллельное проведение формально-абстрактных и содержательно-эмпирических исследований, в силу чего они оказываются весьма поучительными даже в тех случаях, когда дают отрицательные результаты.

ИЗБРАННАЯ БИБЛИОГРАФИЯ (ТЕОРИЯ ФОРМАЛЬНЫХ ГРАММАТИК И ЯЗЫКОВ; ТРАНСФОРМАЦИОННАЯ ГРАММАТИКА) [1])

Основываясь на исследованиях, проводившихся рядом американских структуралистов, в особенности на работах З Хэрриса [1946, 1951, 1952, 1957, 1964], Н. Хомский [1956, 1957] ввел понятие формальной грамматики и определил основные классы формальных грамматик, в дальнейшем формальные грамматики были подвергнуты им (см. Хомский [1959, 1963]) математическому изучению

В то же самое время начала развиваться теория регулярных множеств (регулярных событий), основы которой заложил С. Клини, работы Дж. Майхилла [1957], Н Хомского и Дж Миллера [1958], М Рабина и Д Скотта [1959] и т. д. дали ей новый импульс.

Затем М Шютценберже [1959], введя понятия формального ряда, а также линейных и металинейных языков (в работе [1961b]), положил начало новым интересным исследованиям в области теории автоматов и теории формальных грамматик

Ниже мы укажем лишь самые основные работы, относящиеся к интересующей нас области, впрочем, некоторые из них содержат в свою очередь обширные списки литературы Особо следует отметить монографию С. Гинзбурга [1966], где читатель найдет детальное изложение ряда вопросов, затронутых в настоящей книге.

Бар-Хиллел (Bar-Hillel Y)

[1964] Language and information Selected essays on their theory and application, Academic Press, Addison-Wesley, Jerusalem.
Сборник статей, включающий, в частности, три следующие работы и содержащий список литературы в 155 названий

Бар-Хиллел, Гайфман, Шамир (Bar-Hillel Y, Gaifman C, Shamir E)

*[1960] On categorial and phrase structure grammars, *Bulletin of the Research Council of Israel*, **9F**, 1—16.

Бар-Хиллел, Перлес, Шамир (Bar-Hillel Y, Perles M, Shamir E)

[1961] On formal properties of simple phrase structure grammars, *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, **14**, № 2, 143—172
Подклассы КС-грамматик; нераспознаваемые свойства

Бар-Хиллел, Шамир (Bar-Hillel Y, Shamir E)

[1960] Finite state languages· formal representation and adequacy problems, *Bulletin of the Research Council of Israel*, **8F**, № 3, 155—166
Эквивалентность грамматик и автоматов; проблемы формального описания естественных языков.

---

[1]) В оригинале все работы по трансформационной грамматике были выделены в особый раздел; однако, поскольку многие из них повторялись и в основной библиографии, мы сочли более целесообразным отказаться от такого разделения. Ряд работ добавлен переводчиком и редактором перевода; все такие работы отмечены звездочкой. — *Прим. перев.*

Гинзбург (Ginsburg S)
*[1966] The mathematical theory of context-free languages, New York-San Francisco-Toronto-London. (Перевод: Математическая теория контекстно-свободных языков, изд-во «Мир», М., 1970)

Гинзбург, Партн (Ginsburg S., Partee Barbara)
*[1969] A mathematical model of transformational grammars, *Information and control*, **15**, № 4, 297—334.

Гладкий А В
*[1963] Грамматики с линейной памятью, *Алгебра и логика*, **2**, № 5, 43—55.
*[1965a] Алгоритмическая нераспознаваемость существенной неопределенности КС-языков, *Алгебра и логика*, **4**, № 4, 53—64
*[1965b] Прямое доказательство теоремы Мэтьюза, *Алгебра и логика*, **4**, № 4, 65—70
*[1966] Лекции по математической лингвистике для студентов НГУ, Новосибирск.

Гладкий А. В., Мельчук И. А.
*[1969] Элементы математической лингвистики, изд-во «Наука», М.

Грейбах (Greibach Sheila)
*[1968] A note on undecidable properties of formal languages, *Mathematical systems theory*, **2**, № 1, 1—6.

Гросс (Gross M.)
[1963—1964] Linguistique mathématique et langages de programmation, *Revue française de traitement de l'information*, 231—253
[1966] Applications géométriques des langages formels, *ICC Bulletin*, 5-3.

Дэвис (Davies M)
[1958] Computability and unsolvability, McGraw-Hill, New York.

Диковский А. Я.
*[1969] О соотношении между классом всех контекстно-свободных языков и классом детерминированных контекстно-свободных языков, *Алгебра и логика*, **8**, № 1, 44—64.

Клини (Kleene S. C.)
[1956] Representation of events in nerve nets and finite automata. — In. «Automata studies», Princeton (N-J.), 3—41. (Перевод в сб. «Автоматы», ИЛ, М, 1956, 15—67.)

Клайма (Klima E. S.)
[1965] Current development in generative grammar, Prague colloquium in algebraic linguistics and machine translation, *Kybernetika*, **1**, № 2, 184—197.

Курода (Kuroda S. Y)
[1964] Classes of languages and linear-bounded automata, *Information and control*, **7**, № 2, 207—223

Ландвебер (Landweber P. S)
[1963] Three theorems on phrase structure grammar of type 1, *Information and control*, **6**, № 2, 131—136
В статьях Куроды и Ландвебера устанавливается эквивалентность НС-грамматик и линейно ограниченных автоматов

Майхилл (Myhill J.)
[1957] Finite automata and representation of events — In Myhill J, Nerode A, Tennenbaum S, Fundamental concepts of the theory of systems, 112—137.
Определение полугруппы конечного автомата; стандартные А-языки, А-языки и графы.
[1960] Linear bounded automata, WADD technical note 60—165, Wright Air Development Division
Определение и свойства линейно ограниченных автоматов.

Мак-Нотон, Ямада (McNaughton R., Yamada H.)
[1960] Regular expressions and state graphs for automata, *IRE Transactions on electronic computers*, **EC-9**, № 1, 39—47.
Взаимоотношение между представляющими выражениями и графами конечных автоматов.

Мальцев А. И.
*[1965] Алгоритмы и рекурсивные функции, изд-во «Наука», М.

Мэтьюз (Matthews G. H.)
[1963] Dicontinuity and asymmetry in phrase structure grammars, *Information and control*, **6**, № 2, 137—146. (Перевод в сб. «Математическая лингвистика», изд-во «Мир», М., 1964, 150—159.)
Доказывается эквивалентность некоторого специального класса НС-грамматик (с определенными ограничениями на способ вывода) и класса КС-грамматик. В доказательстве допущена ошибка, исправленная в работе Мэтьюза [1964]; другое доказательство см. в статье Гладкого [1965b].
*[1964] A note on asymmetry of phrase structure grammars, *Information and control*, **7**, № 3, 360—365.

Нерод (Nerode A.)
[1958] Linear automaton transformations, *Proceedings of the American Mathematical Society*, **9**, № 4, 541—544.

Нива (Nivat M.)
[1966] Eléments de la théorie generale des codes, Séminaire OTAN sur la théorie des automates, ed. Caïanello.

Парик (Parikh R.)
[1961] Language-generating devices, *Research laboratory of electronics, Quarterly progress report (MIT)*, № 60, 199—212.
См. также более позднюю публикацию: Parikh R., On context-free languages, *Journal of Association for computing machinery*, **13**, № 4 (1966), 570—580.
Неустранимая неоднозначность КС-языков (определение и пример); другие вопросы теории КС-грамматик.

Пост (Post E.)
[1946] A variant of a recursively unsolvable problem, *Bulletin of American Math. Society*, **52**, 264—268.
Проблема соответствий Поста.

Рабин, Скотт (Rabin M. O., Scott D.)
[1959] Finite automata and their decision problems, *IBM Journal of research and development*, **3**, 115—125. (Перевод: Кибернетический сборник, вып. 4, 1962, стр. 58—91.)

Роджерс (Rogers H.)
[1961] Recursive functions and effective computability, Mass Inst. Tech., Math. Dept. [mimeo], New York, 1967. (Готовится русский перевод.)

Фитиалов С. Я.
*[1968] Об эквивалентности грамматик НС и грамматик зависимостей, см. сб. «Проблемы структурной лингвистики», изд-во «Наука», М., 1967, 71—102.

Хомский (Chomsky N.)
[1956] Three models for the description of language, *IRE Transactions on information theory*, **IT-2**, 113—124. (Перевод: Кибернетический сборник, вып. 2, 1961, 237—266.)
Типы формальных грамматик; лингвистические проблемы, связанные с формальными грамматиками.
[1957] Syntactic structures, Mouton and Co., s'Gravenhage. (Перевод: в сб. «Новое в лингвистике», изд-во «Прогресс», вып. II, 1962, 412—527.)

[1959] On certain formal properties of grammars, *Information and control*, **2**, № 2, 137—167. (Перевод: Кибернетический сборник, вып 5, 1962, стр. 279—312) Математические свойства различных классов формальных грамматик.
[1962] A transformational approach to syntax. — In: «Proceedings of the 1958 Conference on problems of linguistic analysis in English» (Hill A. A., ed.), Austin (Texas), 124—158.
[1963] Formal properties of grammars — In. «Handbook of mathematical psychology», 2 (Luce D., Buch E., Galanter E., eds), New York, John Wiley & Sons, Inc., 323—418. (Перевод: Кибернетический сборник, новая серия, вып. 2, 1966, стр. 121—230.)
Обобщающая статья, содержащая многочисленные результаты относительно свойств автоматов и формальных грамматик. Обширная библиография (78 назв.).
[1965] Aspects of theory of syntax, Cambridge (Mass.).

Хомский, Миллер (Chomsky N., Miller G.)
[1958] Finite state languages, *Information and control*, **1**, № 1, 91—112. (Перевод: Кибернетический сборник, вып. 4, 1962, стр. 231—255.)

Хомский, Шютценберже (Chomsky N, Schutzenberger M P)
[1963] The algebraic theory of context-free languages — In «Computer programming and formal systems» (Brafford P., Hirschberg D., eds), North Holland Publ., Amsterdam, 118—161. (Перевод: Кибернетический сборник, новая серия, вып. 3, 1966, 195—242)
Алгебраические свойства КС-языков; формальные ряды; языки Дика. Обширная библиография.

Хэррис (Harris Z S.)
[1946] From morpheme to utterance, *Language*, **22**, № 3, 161—183
[1951] Methods in structural linguistics, Univ. of Chicago Press, Chicago
См. также более позднюю публикацию этой книги: Structural linguistics, Chicago-London, 1963
[1952] Discourse analysis, *Language*, **28**, № 1, 1—30.
[1957] Co-occurrence and transformations in linguistic structure, Language, **33**, № 3 (Part I), 283—340 (Перевод: в сб. «Новое в лингвистике», вып. II, изд-во «Прогресс», М., 1962, 528—638)
[1964] Elementary transformations, Transformation and discourse analysis papers, № 54, Univ. of Pennsylvania, Philadelphia.
Все указанные здесь статьи Хэрриса перепечатаны в книге: Harris Z S., Papers in structural and transformational linguistics, Dordrecht, Reidel, 1970.

Шейнберг (Scheinberg S)
[1960] Note on the Boolean properties of context-free languages, *Information and control*, **3**, № 4, 372—375.

Шютценберже (Schützenberger M P)
[1959] Un problème de la théorie des automates, Seminaire Dubreil-Pisot, Paris
Определение понятия формального ряда и его использование для исследования автоматов.
[1961a] A remark of finite transducers, *Information and control*, **4**, № 2—3, 185—196.
[1961b] Some remarks on Chomsky's context-free languages, Research laboratory of electonics, Quarterly progress report (MIT), № 63, Cambridge, Mass., 155—170
Определение линейных КС-языков.
[1963a] Certain elementary families of automata. — In: Proceedings of Symposia on mathematical theory of automata, New York, 1962. Brooklin, 139—153.

[1963b] On context-free languages and push-down automata, *Information and control*, **6**, № 3, 246—264
Определение автомата с магазинной памятью, языки Дика

Янгер (Younger D H)
*[1967] Recognition and parsing of context-tree languages in time n^3, *Information and control*, **10**, № 2, 189—209. (Перевод: в сб. «Проблемы математической логики», М., 1970, 344—362.)

ОГЛАВЛЕНИЕ

Предисловие редактора перевода 5
От авторов 7
Предисловие 9

ЧАСТЬ I ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ ИЗ ЛОГИКИ И АЛГЕБРЫ

Глава I. Слова (цепочки). Полугруппы. Языки 13
§ 1.1. Свободная полугруппа 13
§ 1.2. Операции над словами 18
§ 1.3. Языки 23
§ 1.4. Упражнения 26

Глава II. Общие сведения о формальных системах 30
§ 2.1. Описание исчисления высказываний на интуитивном уровне . . 30
§ 2.2. Понятие формальной системы 37
§ 2.3. Формализованный вариант исчисления высказываний 42
§ 2.4. Определение формальной системы 46

Глава III. Комбинаторные системы 49
§ 3.1. Определение комбинаторных систем 49
§ 3.2. Нормальные системы 57
§ 3.3 Упражнения 61
§ 3.4. Независимость от алфавита 61

Глава IV. Алгоритмы. Машины Тьюринга 64
§ 4.1. Алгоритмы 64
§ 4.2. Машины Тьюринга 68
§ 4.3. «Численные» машины Тьюринга 76
§ 4.4. Упражнения 79

Глава V Вычислимость и разрешимость 81
§ 5.1 Исчисление функций 81
§ 5.2 Операции над функциями 83
§ 5.3. Метод Гёделя 86
§ 5.4. Рекурсивные и рекурсивно перечислимые множества 89
§ 5.5 Замечания и дополнения 93

Глава VI. Комбинаторные системы и машины Тьюринга: неразрешимые проблемы 99
§ 6.1. Комбинаторные системы и машины Тьюринга 99
§ 6.2. Неразрешимые проблемы 105

ЧАСТЬ II. НЕКОТОРЫЕ ЗАМЕЧАТЕЛЬНЫЕ КЛАССЫ ЯЗЫКОВ

Глава VII. Контекстно-свободные языки (языки Хомского). общая характеристика и основные свойства 112
§ 7.1. Грамматика и описание синтаксических структур 112
§ 7.2. Определения. Распознаваемые свойства 116
§ 7 3. Свойства замкнутости 123
§ 7.4. Специальные классы КС-языков 126
§ 7.5. Упражнения 129

Глава VIII. Нераспознаваемые свойства КС-грамматик 130
§ 8 1. Проблемы, связанные с пересечением 130
§ 8 2. Проблемы, связанные с неоднозначностью 133
§ 8 3. Существенная неоднозначность минимальных линейных языков 139

Глава IX. Автоматы с магазинной памятью 143
§ 9.1. Автоматы, допускающие языки 143
§ 9.2. Автоматы, порождающие языки 149
§ 9.3. Класс языков, допускаемых (порождаемых) МП-автоматами . 151
§ 9 4. Приложения КС-языков 155

Глава X Автоматные языки и конечные автоматы 159
§ 10 1. А-грамматики 159
§ 10.2. Конечные автоматы 162
§ 10.3. Некоторые обобщения и видоизменения конечных автоматов . 166
§ 10 4. Свойства замкнутости Представление А-языков по Клини . . 168
§ 10 5. А-языки и КС-языки 171
§ 10 6. Односторонние конечные преобразователи 172

Глава XI. Задание языков с помощью систем уравнений 175
§ 11.1. Функции, аргументами и значениями которых являются языки 175
§ 11 2. Языки и формальные степенные ряды 189

Глава XII. Грамматики непосредственно составляющих Автоматы с линейно ограниченной памятью 194
§ 12.1. Грамматики непосредственно составляющих (НС-грамматики) 194
§ 12.2. Линейно ограниченные автоматы 196
§ 12.3 Классификация автоматов 199
§ 12.4 Упражнения 201

ЧАСТЬ III. АЛГЕБРАИЧЕСКИЙ ПОДХОД

Глава XIII. Гомоморфизмы полугрупп 202
§ 13.1. Произвольные полугруппы 202
§ 13 2. Конгруэнция и эквивалентности, сопоставляемые языку . . . 207
§ 13.3. Понятия, связанные с кодами 212

Глава XIV. Дополнительные сведения об автоматных языках 214
§ 14.1. Стандартные А-языки 214

§ 14 2. Свойства стандартных А-языков 217
§ 14.3. Алгебраическое описание А-языков 218
§ 14.4. Преобразования 226

Глава XV. Дополнительные сведения о КС-языках 232
§ 15 1. Языки Дика 232
§ 15.2. Стандартные КС-языки 240
§ 15.3. Совпадение класса КС-языков с классом языков, допускаемых автоматами с магазинной памятью 244
§ 15.4 Упражнения 245

Глава XVI. Алгебраические языки 247
§ 16.1. Дополнительные сведения о формальных степенных рядах . . 247
§ 16 2. Алгебраические ряды 257
§ 16 3 Приложения к языкам 259
§ 16.4. Упражнения 264
§ 16 5. Применение «языковых» уравнений в комбинаторной геометрии 264

ПРИЛОЖЕНИЕ. **ТРАНСФОРМАЦИОННЫЕ ГРАММАТИКИ**

§ 1. Формальные грамматики и естественные языки 272
§ 2. КС-грамматики и трансформации 274
§ 3. Расширение грамматики 275
§ 4. Проблемы, связанные с трансформациями 281
Избранная библиография 287

УВАЖАЕМЫЙ ЧИТАТЕЛЬ!

Ваши замечания о содержании книги, ее оформлении, качестве перевода и другие просим присылать по адресу:

129820, Москва, И-110, ГСП,
1-й Рижский пер., д. 2,
издательство «Мир»

М. Гросс, А. Лантен

ТЕОРИЯ ФОРМАЛЬНЫХ ГРАММАТИК

Редактор *Д. Ф. Борисова*
Художник *К. П. Сиротов*
Художественный редактор *В. И. Шаповалов*
Технический редактор *И. К. Дерва*
Корректор *С. Г. Ефимова*

Сдано в набор 11/III 1971 г.
Подписано к печати 29/IX 1971 г.
Бумага № 2, 60×90¹/₁₆=9,25 бум. л. 18,50 печ. л.
Уч.-изд. л. 16,10. Изд. № 1/5601
Цена 1 р. 66 к. Зак. 1036

ИЗДАТЕЛЬСТВО «МИР»
Москва, 1 й Рижский пер., 2

Ордена Трудового Красного Знамени
Ленинградская типография № 2
имени Евгении Соколовой Главполиграфпрома
Комитета по печати при Совете Министров
СССР. Измайловский проспект, 29