МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА

Факультет вычислительной математики и кибернетики

УДК 519.682.1+510.53

Л.И.Станевичене

К ТЕОРИИ БЕСКОНТЕКСТНЫХ ЯЗЫКОВ

Разрешаю депонирование в ВИНИТИ

Зам.декана факультета вычислительной
математики и кибернетики МГУ
профессор В.М.Пасконов

Подпись автора:

Москва

2000

Депонированная рукопись

УДК 519.765

Dғрафы в теории магазинных автоматов / Вылиток А.А., Станевичене Л.И., Чернцов И.В.; МГУ. -М., 1996. -64с. -Библиогр. 26 назван. -Рус. -Деп. в ВИНИТИ

Дана характеризация КСязыков так называемыми Dғрафами. Представлены преобразование магазинного автомата в эквивалентный D-граф и решения ряда задач, использующие возможность такого преобразования.

Предложен алгоритм построения по магазинному автомату эквивалентной КСғрамматики, которая наследует свойства однозначности, детерминированности и др.

Выделен класс сингулярных магазинных автоматов, которые можно считать аналогами КСғрамматик без самовставления. Доказано, что сингулярные магазинные автоматы допускают в точности регулярные языки.

Исследуются магазинные автоматы с одним поворотом, действующие в реальное время. Приводятся отвечающие данному случаю решения некоторых алгоритмических проблем, в частности, проблемы эквивалентности.

Доказано, что произвольный детерминированный магазинный автомат эквивалентен детерминированному же автомату, каждый такт которого, не стирающий в магазине, читает входной символ. Таким образом, затронут важный для приложений вопрос повышения эффективности детерминированного магазинного автомата.

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им.М.В.ЛОМОНОСОВА

Факультет вычислительной математики и кибернетики

Станевичене Лариса Ивановна

УДК 519.682.1+510.53

К ТЕОРИИ БЕСКОНТЕКСТНЫХ ЯЗЫКОВ

Москва 2000

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Теория формальных языков возникла во второй половине пятидесятых годов. Она использовала для целей описания языков уже существовавший в математике формальный аппарат. В работах Н. Хомского [Chomsky 56], [Chomsky 57], [Chomsky 59a] были выделены классы специальных исчислений, названных им грамматиками, играющие важную роль в современных приложениях теории формальных языков.

Особо важным среди выделенных классов является класс сравнительно простых бесконтекстных грамматик, которые называют также контекстноєвободными. Весьма распространено употребление аббревиатуры "КСѓрамматика". Но термин "бесконтекстная грамматика"кажется нам более удачным стилистически. Изучение именно бесконтекстных грамматик обусловило своеобразное врастание теории формальных языков в информатику.

Теория формальных грамматик сразу же сомкнулась с теорией автоматов. В начале шестидесятых годов было формализовано понятие магазинного автомата [Oettinger 61], [Schutzenberger 63] и установлено, что язык является бесконтекстным тогда и только тогда, когда он допускается магазинным автоматом [Chomsky 62], [Evey 63]. Отметим, что полезность магазинов была понята еще в пятидесятых годах [Burks -Warren -Wright 54], [Newell -Shaw 57], [Ершов 58].

Потребности теории и практики перевода языков программирования обусловили наибольший интерес к подклассам магазинных автоматов, обладающих свойствами однозначности и детерминированности. Согласно [Ginsburg 66] однозначные магазинные автоматы введены в [Haines 65]. Детерминированные магазинные автоматы, хотя и в особой форме и под другим названием, впервые изучались в [Schutzenberger 63]. В частности, там было доказано, что детерминированный, т. е. допускаемый детерминированным магазинным автоматом, бесконтекстный язык является однозначным. Многие свойства детерминированных магазинных автоматов и детерминированных языков были исследованы в [Ginsburg -Greibach 65], [Ginsburg -Greibach 66].

Работа по переводу (трансляции) языков программирования вызвала интерес к формальному определению перевода, т. е. отношения, связывающего с каждой цепочкой языка другую цепочку — результат перевода. В [Feldman -Gries 68] высказывается мнение, что с развитием теории перевода программирование начало возвращать долг теории формальных языков. С понятиями бесконтекстной грамматики и конечного и магазинного автоматов связаны соответственно два фундаментальных метода определения перевода: с помощью так называемой схемы перевода и с помощью преобразователя. К пионерским работам о первом из методов, подругому, о синтаксически управляемой трансляции, относятся [Irons 61] и [Barnett -Futrelle 62].

Первое приближение к современному понятию конечного преобразователя было дано в работе [Ginsburg 62]. Затем оно последовательно обобщалось в работах [Chomsky 62] и [Elgot -Mezei 65]. Преобразователь с магазинной памятью — это магазинный автомат, который снабжен выходной лентой и, разумеется, способностью писать на ней и о котором говорят, что преобразователь с ним согласован [Ginsburg 66] или что он лежит в основе преобразователя [Aho -Ullman 72]. Результат перевода пишется на выходной ленте слева направо по мере чтения

входной цепочки. Понятие преобразователя с магазинной памятью, используемое и в нашей работе, впервые формализовано в [Evey 63].

Отметим, что синтаксический анализ можно рассматривать как перевод, при котором порождаемые бесконтекстной грамматикой цепочки отображаются в некоторые представления их деревьев вывода. Понятие дерева вывода, издавна применявшееся в лингвистике, можно найти во многих источниках под различными названиями. В явной форме оно содержится в [Bar-Hillel -Perles -Shamir 61]. Среди представлений дерева вывода популярны левый или правый выводы и разбор, определяемый, например, как обращение правого вывода. Заметим, что идея "дисциплинированного"вывода появилась в [Evey 63]. Точнее, там рассматривался левый вывод. Обсуждаемые в нашей работе преобразователи с магазинной памятью сопоставляют входным цепочкам их разборы.

Очень скоро стало понятным, что класс детерминированных языков содержит довольно обширные и удовлетворительно отражающие синтаксические черты языков программирования подклассы, в случае которых можно строить анализаторы (т. е. алгоритмы синтаксического анализа), расходующие на обработку цепочки длины n время $c_1 n$ и память $c_2 n$, где c_1 и c_2 — малые константы. Среди методов синтаксического анализа, рассчитанных на такого рода подклассы детерминированных языков, большую популярность приобрели методы предшествования, публикации о которых появились в начале шестидесятых годов [Floyd 63], [Pair 64].

Методы предшествования изучались и обобщались многими исследователями. Наиболее полная, повидимому, библиография по этому вопросу имеется в [Aho -Ullman 72], [Aho -Ullman 73]. Между прочим, работы [Floyd 63] и [Wirth -Weber 66] (последняя излагает полученный независимо от [Pair 64] метод простого предшествования) произвели на нас большое впечатление и повлияли на наше восприятие других методов синтаксического анализа. В [Гончарова 75], [Станевичене 76], [Станевичене 78], [Stanevichene 79], [Миронова -Станевичене 82] было, в частности, формализовано определение "добавки"к методу простого предшествования, необходимой для получения LR(k)-метода [Knuth 65]. Полученная вариация LR(k)-метода применима на практике. Опыт применения описан в [Изимбетов 88], [Миронова 92].

В 1965 году появилась знаменитая статья [Knuth 65], в которой было обнаружено, что время и память, пропорциональные длине анализируемой цепочки, достаточны в случае любого детерминированного языка. Она вызвала многочисленные попытки усовершенствовать или обобщить предложенный в ней механизм синтаксического анализа детерминированных языков. Интерес к этой теме не остыл до сих пор, как свидетельствует, например, статья [Lee -Choe 94]. Однако отложим обсуждение этого направления исследований, чтобы указать другие направления, намеченные основоположниками теории формальных языков.

В ранних работах по теории формальных языков [Chomsky 62], [Chomsky 63], [Chomsky -Schutzenberger 63] приведено любопытное наблюдение о возможности характеризовать языки морфическими образами множеств, в бесконтекстном случае определяемых с помощью языков Дика. Обзор результатов, имеющих аналогию с теоремой Хомского-Шютценберже о представлении каждого

бесконтекстного языка морфическим образом пересечения языка Дика с локальным (в [Chomsky -Schutzenberger 63] локальный язык назван стандартным регулярным событием), приводится в [Hirose -Okawa -Yoneda 85]. Некоторые результаты о представлении языков посредством морфизмов отнесены к математическим жемчужинам [Salomaa 81].

Отметим также характеризации бесконтекстных языков системами уравнений и специфическими выражениями, указывающими на стремление авторов разработать схемы исследования бесконтекстных языков, аналогичные успешно примененным в теории регулярных языков. К наиболее ранним зарубежным работам по этой теме относится [Intema 67]. Некоторые последующие работы переведены на русский язык, например, [Gruska 71], [McWhirter 71]. В тот же период тема разрабатывалась отечественными учеными; так, в [Гладкий 73] рассматриваются подстановочные выражения, теория которых восходит к работам [Диковский 72а], [Диковский 72б].

Система уравнений для задания бесконтекстного языка [Летичевский 69] является результатом алгебраической трактовки агазинного автомата. Заметим, что такая система легко усматривается в нашем так называемом ядре магазинного автомата.

Теория формальных грамматик и языков рассматривает некоторые алгоритмические проблемы, что естественно для дисциплины, возникшей из потребностей лингвистики и информатики. Многие алгоритмические проблемы оказались неразрешимыми, что было установлено к середине шестидесятых годов. Повидимому, самая ранняя публикация в этом ряду — это работа [Bar‑Hillel - Perles -Shamir 61], в которой доказана неразрешимость проблемы эквивалентности бесконтекстных грамматик — весьма важный факт, заставляющий применять для преобразования языковых описаний лишь процедуры, которые заведомо дадут эквивалентный результат. Еще один из впечатляющих (по крайней мере, на наш взгляд) фактов этого ряда — нераспознаваемость неоднозначности бесконтекстных языков, независимо доказаннная в [Гладкий 65б] и [Ginsburg -Ullian 66].

Таким образом, в шестидесятые годы бурно шло накопление результатов теории бесконтекстных языков. Книга [Гладкий 73] и другие источники свидетельствуют, что советские ученые уже в этот период внесли большой вклад в теорию формальных языков; яркое явление — работы А.В.Гладкого и А.А.Летичевского, например. Тогда же были поставлены вопросы, и сейчас остающиеся актуальными. Они связаны с построением алгоритмов, которые обслуживали бы потребности приложений, заинтересованных в автоматическом преобразовании языковых описаний достаточно широкого класса, в проверке эквивалентности таких описаний. Отсутствие приемлемых алгоритмов вынуждает к использованию наиболее простых формальных языковых моделей и к преодолению возникающих при этом трудностей специальными для каждого случая приемами.

Наличие трудных мест в теории бесконтекстных языков связано, повидимому, с тем, что некоторая специфика этих языков недостаточно четко выражена в изучаемых теорией классических формализмах. Вряд ли можно, например, считать,

что техника анализа поведения магазинного автомата, потребная при рассмотрении проблем эквивалентности и регулярности в классе детерминированных языков, в нужной мере разработана: опубликована не одна работа по указанной теме, не имеющая необходимого уровня строгости и ясности и даже ошибочная.

В свете сказанного актуальным является поиск таких характеризаций бесконтекстных языков, которые дали бы понятийную среду, удобную для выражения и исследования явлений, связанных с трудными задачами теории бесконтекстных языков.

Одна из целей настоящей работы как раз и состоит в создании характеризации бесконтекстных языков, наиболее удобной для развития их теории. В огромном богатстве наблюдений, накопленных в теории формальных языков и в прикладных работах, не могли отсутствовать указания, которые, впрочем, стали для нас заметны только тогда, когда цель и средства ее достижения в основном оформились.

Начальный этап работы был подготовлен нашим участием в разработке и реализации языков программирования, которое отражено в работах [Гончарова 71], [Гончарова 72], [Гончарова - Станевичюс 71] и других, определенным образом подытоженных в [Бурова -Станевичене -Станевичюс -Шкляр 83]. Эти занятия пробудили большой интерес к упоминавшимся выше методам синтаксического анализа, поддержанный С.С.Лавровым. Именно наблюдение специфики детерминированных магазинных автоматов, скрытых за LR(1)-таблицами, вселило уверенность в возможность создания наиболее плодотворного инструмента анализа магазинных автоматов и бесконтекстных языков.

Здесь мы даем не полный обзор результатов, которые можно назвать подготовляющими нашу работу, а скорее историю (также не полную) наших встреч с подсказками и подтверждениями правильности выбранного пути.

Одна из первых наших идей — получить графы, которые эквивалентны магазинным автоматам и естественным образом обобщают то, что теперь называют иногда просто конечными автоматами [Salomaa 81], [Eilenberg 74] (в начальный период формирования терминологии бытовали более громоздкие обороты, например, "граф для диаграммы состояний (строго конечного) автомата"[Chomsky 63]). Такие графы давали бы зримый образ вычислений магазинного автомата во всех их деталях, заставили бы понятия теории графов больше работать в задачах теории формальных языков. Была и некоторая надежда (очень слабая, так как привлечение магазина вносит, очевидно, новое качество), что графическое представление магазинных автоматов поможет естестенным образом обобщить на бесконтекстный случай какие-либо достижения теории регулярных языков. Надежда эта не была совершенно лишена основания: можно заметить, насколько родственны по приемам доказательства теорема о представлении регулярного языка морфическим образом локального языка [Salomaa 81] и теорема 6 главы 1 данной работы. Следует также отметить, что в параграфе 4 главы 3 продемонстрирована возможность привлечения приемов, использованных, например, в [Eilenberg 74] для изучения конечных автоматов, для исследования графов магазинных автоматов реального времени с одним поворотом.

Нечто похожее на наш подход можно найти и у других исследователей (см., например, "подход диаграмм состояний для магазинных автоматов"в [Salomaa -

Wood - Yu 94]). Но применение этих разработок обычно невелико; иногда оно ограничивается новыми доказательствами леммы Огдена.

Для обеспечения стандартности и лаконичности информации, определяющей дугу графа магазинного автомата, требовалась некоторая нормальная форма автомата. Была использована форма, характеризуемая следующей дисциплиной работы с магазином: 1) каждый такт или добавляет в магазин ровно один символ, или стирает, также ровно один символ, никогда не стирая исходного содержимого магазина — так называемого маркера дна; 2) к концу чтения допускаемой автоматом цепочки он стирает все, что записывал, при этом магазин не опустошается: в нем остается маркер дна. Черты данной формы магазинного автомата можно найти в большом числе работ. Наиболее близка нашей форма, рассмотренная А.В.Гладким; да и техника протоколов (о них см. в [Гладкий 65а], [Гладкий 73]) ассоциируется с нашим понятием маршрута на графе магазинного автомата.

Поиск подходящей формы магазинного автомата явился и поиском уравновешенности двух средств запоминания информации о допускаемых цепочках: состояний и магазина, одинаковой востребованности их во всех конфигурациях, возникающих при обработке правильной входной цепочки. Ненормальны, например, такие записи в магазине, которые станут "мусором"в заключительной конфигурации. Вряд ли нормально и опустошение изначально непустого магазина, в результате чего автомат "выключается".

Аккуратное соответствие записей в магазин и стираний, кажущееся необходимым (оно, конечно, не достаточно) в удачно построенном магазинном автомате, живо напоминает о скобках. С другой стороны, парные скобки — это лишь модель парных символов и даже цепочек, присутствие которых отличает бесконтекстные языки, не являющиеся регулярными.

Понятия скобки и скобочной системы формализованы с помощью языков Дика, самые полные сведения о которых среди легко доступных нам источников содержит книга [Lallement 79]. Алгебраическая интерпретация языков Дика побудила нас подняться немного над теорией магазинных автоматов и построить обобщение ограниченных языков Дика [Lallement 79], более точно отражающее характер объектов, парных в цепочках бесконтекстных языков. Далее будем называть ограниченные языки Дика просто языками Дика, следуя традиции, сложившейся в работах по теории формальных языков.

Название "Dязыки"предлагаемого обобщения языков Дика сохраняет в себе первую букву словосочетания "Dyck languages". Данная работа базируется на систематическом использовании понятия Dязыка.

В нашем обобщении пара скобок — это пара символов, как и в случае языков Дика. Попрежнему, множества открывающих и закрывающих скобок не пересекаются. Но теперь в различных парах открывающие или закрывающие элементы могут совпадать. Случай парных цепочек, длина хотя бы одной из которых не равна единице, не отражен здесь непосредственно. Однако вычисления магазинных автоматов, "аккуратно"использующих магазин, имеют теперь полную аналогию со скобочными системами.

В данной работе введены термины для определенных фрагментов скобочных систем. Формализованы представления о глубине и ширине — числе последовательно сцепленных скобочных подсистем. Доказана конечность числа скобочных

систем с ограниченными глубиной и шириной. Все это способствовало строгости, ясности и краткости в изложении наших результатов.

Подобную "теорию скобочных систем", связанную с языками Дика, мы обнаружили в [Ehrenfeucht -Hoogeboom -Rozenberg 86]. Однако важность понятия скобки отмечалась во многих работах, в том числе прикладных. Очень убедительно писал о роли скобок Леви [Lévy 75]. Отметим, что работа [Lévy 75] стимулировала ряд исследований на тему нейтрализации синтаксических ошибок. В одной из работ этого ряда сделана попытка обобщить понятие скобки [Ciesinger 76].

В работе [Ehrenfeucht -Hoogeboom -Rozenberg 86] подчеркивается полезность изучения комбинаторных свойств скобочных систем, буквально:

> We formulate and prove a number of combinatorial properties of Dyck words ... Since Dyck words are used in the investigation of various types of data structures, these results seem to be of independent (combinatorial) interest.

В ней, в частности, определяются понятия глубины и ширины цепочки языка Дика и выводится верхняя оценка длины цепочки через ее глубину и ширину. Она того же порядка, что и наша. С помощью указанных результатов работа [Ehrenfeucht - Hoogeboom - Rozenberg 86] изучает представление магазинного автомата парой согласованно применяемых грамматик. Грамматики порождают: одна — читаемые автоматом входные цепочки, другая — соответствующие магазинные цепочки. Последняя грамматика не вполне традиционна: она не различает терминальных и нетерминальных символов.

Определение графа магазинного автомата и связанная с ним система понятий опубликованы в [Станевичене 83], [Станевичене 84], [Станевичене 87a], [Станевичене -Мельников 88], [Станевичене 89], [Кузнецова -Ожиганов -Сагинтаева -Станевичене 90], [Станевичене 93a], [Станевичене -Вылиток 93], [Stanevichene 94], [Вылиток -Станевичене -Чернцов 96] и [Станевичене 99]. Но терминологию можно считать сложившейся только в двух последних работах. В них впервые дана характеризация бесконтекстных языков Dғрафами.

В системе понятий, основанной на нашей графической интерпретации бесконтекстных языков, понятие ядра является, возможно, главнейшим. Понятие циклического пути вместе с понятием скобочной системы помогло нам осмыслить, что некоторое конечное подмножество путей на Dғрафе (аналогично, некоторое конечное подмножество сентенциальных форм бесконтекстной грамматики) выражает закон образования цепочек языка из подцепочек той его части, которая определяется выделенным подмножеством — ядром.

Упомянутый закон, повидимому, проще всего объяснить как закон получения цепочек, отвечающий так называемым КСвыражениям, которые определяются в главе 1. В этом определении вместо регулярной операции итерации рассматривается новая операция — именование. Эффект операции именования определен с помощью понятия скобки. На наш взгляд, в понятии КСвыражения концентрируются общие черты изучаемых в данной работе формализмов для описания бесконтекстных языков (магазинных автоматов, бесконтекстных грамматик, D-графов), чего нельзя сказать о других известных выражениях для задания этих

языков. Крме того, не уступая известным в лаконичности, КСвыражения подробней и наглядней указывают закон образования цепочек языка из "подчиненных"цепочек. КСвыражения обобщают регулярные [Kleene 56]; есть надежда, чт они приближают нас к адекватной алгебраической трактовке магазинных автоматов и бесконтекстных грамматик.

Легко вообразить, что между магазинными автоматами и бесконтекстными грамматиками существует соответствие (назовем его сходством), при котором ядра соответствующих объектов определяют одно и то же бесконтекстное выражение. В связи с этим возникает надежда на существование практически полезных алгоритмов, преобразующих одни описания бесконтекстного языка в другие. К полезным следует отнести алгоритм преобразования бесконтекстной грамматики в магазинный автомат, не слишком громоздкий по сравнению с исходной грамматикой (ср. задачу уменьшения LR(k)анализаторов, о которой речь пойдет ниже).

Построение по магазинному автомату схожей бесконтекстной грамматики рассмотрено А.А.Вылитком [Станевичене -Вылиток 93], [Вылиток -Станевичене - Чернцов 96], [Вылиток 98]. В последней работе доказано, что в этой грамматике сохраняется однозначность, детерминированность (в виде LR(1)євойства), одноповоротность (в форме линейности), неспособность к согласованной итерации двух непустых цепочек и некоторые другие свойства исходного автомата. Заметим, что А.А.Вылиток использует лишь подмножество ядра магазинного автомата, достаточное для построения схожей грамматики. На самом деле этого подмножества достаточно, повидимому, и в большинстве задач, но допущенная в определении ядра избыточность облегчает доказательство некоторых теорем. Заметим еще, что в [Вылиток 96] получена верхняя оценка длин путей, рассмотрение которых требуется для построения ядра Dғрафа, существенно лучшая, чем приведенная в настоящей работе. Основной фактор, обеспечивший улучшение оценки — возможность рассматривать входящие в ядро пути не целиком, а по некоторым их участкам.

Отметим теперь, что кнутовские LR(1)ҭаблицы — это представление преобразователя с магазинной памятью, который сопоставляет входной цепочке ее разбор относительно грамматики, определившей таблицы. При оптимизации такого преобразователя необходима схожесть старого и нового преобразователей, чтобы исходный синтаксис не был "забыт". Так как только требование схожести и ограничивает в выборе средств оптимизации, перебор преобразователей некоторого конечного множества позволяет найти минимальный, например, по числу состояний (кстати, уменьшение LR(1)ҭаблиц, предлагаемое в известных работах, обычно можно интерпретировать как уменьшение числа состояний обсуждаемых здесь преобразователей).

Если исключить грамматики некоторого "вырожденного", специально придуманного, вида, то для обычно применяемых в практике трансляции бесконтекстных грамматик число перебираемых вариантов будет не слишком велико. Это обусловлено следующим.

Некоторые особенности LR(k)ҭаблиц, заведомо обеспечивающие их уменьшение, были замечены многими исследователями. Были даже выделены подклассы LR(k)ғрамматик, имеющих LR(k)анализаторы сравнительно малого объема. Наиболее известны подклассы SLR(k)- и LALR(k)ғрамматик, определенные в

[DeRemer 69], [DeRemer 71]. Интересно, что распространение идеи LR(k)-метода на грамматики более общих классов (кроме наших работ, упоминавшихся выше, и [Lee -Choe 94] укажем [Walters 70] и [Шумей -Зонис 75], где рассматриваются контекстные грамматики), также может быть связано с приемами уменьшения таблиц анализатора. Выработанная техника уменьшения LR(k)-таблиц может быть обобщена в совокупности условий, которая отсеет подавляющую часть безуспешных вариантов.

В данной работе (см. также [Станевичене 96б]) вопрос уменьшения LR(k)-таблиц решается как подзадача более общей задачи. Мы указываем метод синтаксического анализа бесконтекстных языков, реализуя идею схожести синтаксических описаний и распространяя идею LR(1)-метода на случай всех бесконтекстных грамматик. Наш метод имеет ту общую черту с известным алгоритмом Эрли [Earley 68], что дает тем более эффективный анализатор, чем "лучше"грамматика. В частности, если грамматика принадлежит LR(1)-классу, то наш метод обеспечивает построение LR(1)-анализатора. Вообще, анализатор отражает все черты исходной бесконтекстной грамматики, так как он строится по ее ядру и может быть представлен преобразователем с магазинной памятью, который основан на магазинном автомате, схожем с заданной грамматикой. В случае праволинейной грамматики соответствующий преобразователь легко превращается в конечный.

Итак, в нашей работе суммируются и усиливаются результаты исследований LR(k)-метода, не прибегающих к обработке грамматики "по частям". Но мы считаем очень важной идею работы [Korenjak 69], в которой рассматривались эвристические приемы разбиения грамматики с тем, чтобы построить LR(k)-таблицы по частям, каждая из которых не слишком велика.

Идея Кореньяка развита в работах [Игнатов 86], [Игнатов 87а], [Игнатов 87б], выполненных под нашим руководством. Там построен формальный алгоритм разбиения грамматики, управляемый отношением, определяемым также формально и указывающим группы нетерминальных символов, разъединение которых по разным подграмматикам нежелательно. Определение данного отношения выражает фактически некоторые особенности магазинного автомата, схожего с грамматикой (в смысле, указанном выше) и лежащего в основе LR(k)-анализатора.

В нашей работе идея расщепления языкового описания с целью обработки по частям получила дальнейшее развитие. В главе 2 приводятся алгоритмы преобразования D-графа в КС-выражение и конечного автомата в регулярное выражение, базирующиеся на отщеплении частей обрабатываемых графов, которые (части) определяются с помощью понятия сильно связной компоненты графа.

По нашему мнению, прием расщепления грамматики вместе с нашими приемами уменьшения анализаторов должен быть положен в основу программы построения эффективных анализаторов детерминированных языков.

Предыдущее обсуждение показывает полезность нашего подхода, применяющего понятия D-графа и КС-выражения. Рассмотрим еще два примера, когда наш подход обеспечивает заметные результаты. Они касаются трудных алгоритмических проблем: регулярности и эквивалентности в классе детерминированных языков. Эти проблемы были впервые поставлены в [Ginsburg -Greibach 65]. В статьях [Stearns 67], [Valiant 75] и др. излагаются доказательства разрешимости проблемы регулярности языка, допускаемого детерминированным магазинным автоматом. В статьях [Мейтус 92] и [Senizergues 97] представлены два доказательства

алгоритмической разрешимости проблемы эквивалентности детерминированных магазинных автоматов.

В работе [Stearns 67] изложение ведется, по словам автора, "на слегка неформальной основе"с целью облегчить чтение. Это решение автора вряд ли удачно: работа непонятна (обсуждение некоторых ее неясностей приводится в главе 3 настоящей работы), и возникает сомнение не только в правильности приведенных в ней оценок, но даже и в разрешимости рассматриваемой проблемы.

Последующие работы направлены на уменьшение сложности алгоритма проверки регулярности детерминированного языка. К сожалению, и они не свободны от пробелов и неясностей, а работа [Shankar -Adiga 91, 92], за рубежом признанная, повидимому, как наиболее успешная, на самом деле ошибочна. Ошибочен алгоритм проверки регулярности детерминированного языка, основанный на убеждении авторов в отсутствии LR(1)ѓрамматик с самовставлением, порождающих регулярные языки. (Впрочем, допустимо и то, что они убеждены в обратном, но не разглядели законов взаимосвязи LR(1)ѓрамматик с изучаемыми графами изза громоздкости последних. По существу, в [Shankar -Adiga 91, 92] рассматриваются наши Dѓрафы, но очень специфические и большие в случае даже самых простеньких грамматик. Эти графы описывают переходы LR(1)ѓанализатора и более обширны, чем LR(1)ѓаблицы, так как, в отличие от таблиц, раскрывают технику свертки по правилу грамматики.) Пример с языком $\{a^m b a^n b | m \geq 1,\ n \geq 0\}$ в нашей главе 3 доказывает, что они заблуждаются. Вариации примера опубликованы в нескольких наших работах, достаточно давних, в том числе в "Докладах РАН"(см. [Станевичене 95]), которые переводятся на английский язык. Но, повидимому, зарубежные читатели не заметили этой публикации или не сделали из нее надлежащих выводов.)

В нашей работе определяются праволинейные магазинные автоматы, рассмотренные в [Станевичене - Чобан 94] и [Stanevichene - Choban 94] под другим названием. Об автоматах этого подкласса можно сказать, что они аналогичны праволинейным грамматикам, которые являются частным случаем грамматик без самовставления, введенных Н.Хомским [Chomsky 59a], [Chomsky 59b]. Праволинейный автомат допускает регулярный язык. Конечный автомат легко интерпретировать как праволинейный магазинный автомат частного вида, поэтому справедливо и то, что любой регулярный язык допускается некоторым праволинейным магазинным автоматом. Таким образом, имеем еще одну характеристику регулярных языков.

В работе [Вылиток 98] найден более общий класс магазинных автоматов, характеризующий регулярные языки.

Мы устанавливаем, что праволинейность магазинного автомата распознается по его ядру (причем в детерминированном случае достаточно лишь сравнительно просто устроенных элементов ядра), и получаем достаточное условие регулярности бесконтекстного языка. В главе 3 на этой основе очень просто формулируется частичный алгоритм проверки регулярности бесконтекстного языка.

Проблема эквивалентности детерминированных магазинных автоматов долгое время сопротивлялась решению; ей посвящено очень много работ. Кроме работы [Мейтус 88], слишком небрежно написанной, чтобы ктоѓто мог ее понять, разрешимость проблемы эквивалентности детерминированных магазинных автоматов

утверждают статьи [Мейтус 92] и [Sénizergues 97]. Они не полны, содержат опечатки и ошибочные вспомогательные утверждения (но влияние этих ошибок на правильность основного результата не ясна). В настоящий момент нам не удалось преодолеть все эти трудности и понять, почему их авторы правы. По нашим сведениям упомянутые статьи непонятны и многим другим людям, даже высококвалифицированным математикам.

Существуют работы, которые доказывают разрешимость проблемы эквивалентности для детерминированных магазинных автоматов или бесконтекстных грамматик частного вида, не охватывающих весь класс детерминированных языков (см., например, [Анисимов 77], [Зубенко 78], [Мейтус 89], [Непомнящая 81], [Романовский 80], [Романовский 86], [Яффе 74], [Korenjak -Hopcroft 66], [Nepomnjashchaja 84], [Oyamaguchi 87], [Valiant 73], [Valiant 74], [Yair -Amiram 84], [Tomita 95] и ссылки в перечисленных работах). Есть и работы, которые предприняты в связи с проблемой эквивалентности детерминированных магазинных автоматов и содержат алгоритмы проверки эквивалентности устройств, определяющих не все детерминированные языки, но зато определяющих и некоторые недетерминированные языки. К таковым относится выполненная под нашим руководством работа Б.Ф.Мельникова.

Проблема эквивалентности некоторых минимальных линейных грамматик (могущих порождать не только детерминированные языки), рассматриваемая в [Мельников 89], [Мельников 90] и [Melnikov 93], связана с наблюдением, которое касается поведения так называемых парных циклов на Dғрафе.

В статьях [Станевичене 93б], [Станевичене 95], [Станевичене 96а] получен результат, который играет направляющую роль для математиков, занятых построением алгоритмов, которые решают вопрос об эквивалентности детерминированных магазинных автоматов. В них рассмотрена правомерность такого подхода к решению, когда для проверки эквивалентности детерминированных магазинных автоматов имитируется их совместная работа. Идея имитации очень естественна и использовалась некоторыми авторами при рассмотрении частных случаев проблемы. Повидимому, первой работой такого рода является [Valiant 73], где предложена техника имитации и доказана с ее помощью разрешимость проблемы эквивалентности детерминированных магазинных автоматов с конечным числом поворотов. В статьях [Романовский 80], [Романовский 86] эта техника обобщается. К сожалению, приведенное в [Романовский 86] доказательство разрешимости проблемы эквивалентности детерминированных магазинных автоматов, действующих в реальное время, ошибочно (по этому поводу также см. [Станевичене 93б], [Станевичене 95], [Станевичене 96а]), а из работы [Чернцов 95], усиливающей наш результат, следует, что на выбранном в [Романовский 86] пути невозможно добиться успеха.

В [Станевичене 93б], [Станевичене 95], [Станевичене 96а] вводится в терминах графов магазинных автоматов понятие сопряжения, формально определяющее совокупность сведений, необходимых имитирующему устройству для прослеживания одновременной работы двух магазинных автоматов над одной и той же входной цепочкой. Затем определяется понятие графа сопряжений двух магазинных автоматов, который является подграфом некоторого конечного графа, легко создаваемого по заданным автоматам. Граф сопряжений, по существу, представляет собой квинтэссенцию всех возможных имитирующих устройств. Понятие

графа сопряжений сходно в глубинной своей основе с понятием пересечения (или произведения) конечных автоматов, которое используется иногда для доказательства замкнутости класса регулярных языков относительно операции пересечения.

Устанавливается несуществование алгоритма, который по любым двум эквивалентным детерминированным магазинным автоматам строит их граф сопряжений. Оказалось, что существование такого алгоритма означало бы и существование алгоритма, решающего проблему соответствия Поста [Post 47].

Отметим, что проблема соответствия Поста широко применяется в теории формальных языков; одно из первых наших упражнений в применении этой проблемы излагается в [Станевичене 87б]. Недаром сравнительно простое доказательство ее неразрешимости дано Р.Флойдом в [Floyd 64], сделавшим большой вклад в теорию перевода языков программирования. Он же сформулировал проблему частичного соответствия — вариант проблемы соответствия Поста, послуживший, например, для доказательства неразрешимости проблемы вхождения бесконтекстной грамматики в LR(k)класс для какогонибудь $k \geq 0$ [Knuth 65]. Видоизменения постановки проблемы соответствия Поста можно найти в [Ruohonen 83].

В главе 3 данной работы кроме проблемы построения графа сопряжений рассматривается еще одна алгоритмическая проблема, из разрешимости которой следовала бы разрешимость проблемы эквивалентности детерминированных магазинных автоматов: проблема эквивалентности памятей (она впервые сформулирована в [Stanevichene 95]). Памятью названа пара

(состояние, непустая цепочка магазинных символов),

которая входит в состав некоторой конфигурации магазинного автомата. При этом второй элемент пары является, вообще говоря, лишь некоторой верхней частью содержимого магазина. Проблема эквивалентности памятей неразрешима для детерминированных магазинных автоматов.

В главе 3 указаны и положительно решаемые проблемы как примеры изложения известных фактов в наших терминах. Тут привлекательна лаконичность доказательств.

Один из параграфов главы 3 предлагает технику проверки эквивалентности магазинных автоматов с одним поворотом, действующих в реальное время, сравнимую по эффективности с техникой, разработанной для конечных автоматов. Успешный перенос на рассматриваемый случай методов, разработанных в теории регулярных языков, обусловлен использованием нашего подхода к описанию бесконтекстных языков.

Итак, работа предлагает новый подход в теории бесконтекстных языков, основанный на обобщении языков Дика, которое, по сравнению с последними, удачнее отражает особенности парных объектов, присутствующих в цепочках нерегулярных бесконтекстных языков. Этот подход позволяет выявить внутреннюю общность различных характеризаций бесконтекстных языков и получить ответы на достаточно трудные теоретические вопросы. В подтверждение действенности подхода рассмотрен ряд теорем. К основным можно отнести следующие результаты работы.

1. Предложены способы описания бесконтекстных языков так называемыми D-графами и КС-выражениями. На базе этих формализмов разработана система понятий, плодотворная в исследовании вопросов теории бесконтекстных языков.

2. Разработан аппарат исследования магазинных автоматов, который обеспечивает в некоторых специальных случаях весьма эффективную проверку эквивалентности автоматов и регулярности допускаемого автоматом языка.

3. Поставлены алгоритмические проблемы, формализующие некоторые подходы к построению алгоритма, проверяющего эквивалентность детерминированных магазинных автоматов. Доказана неразрешимость этих проблем в довольно узких подклассах детерминированных магазинных автоматов.

4. Предложен метод построения по бесконтекстной грамматике эквивалентного магазинного автомата, свойства (однозначность, детерминированность и т. пр.) и размеры которого определяются свойствами и размерами исходной грамматики. На этой основе создан метод построения недетерминированного анализатора, который в каждом из случаев разбора входной цепочки работает как LR(1)-анализатор, и получено теоретическое решение вопроса оптимизации LR(1)-таблиц.

5. Найдена методика расщепления D-графов, которая опирается на результаты теории графов и позволяет активнее использовать прием разделения в практике обработки языковых описаний.

Не определяемые в работе термины и обозначения заимствованы из книги [Ginsburg 66]. Они большей частью повторяются и в других широко распространенных источниках. Термин "алфавит"применяется к непустому конечному множеству. Пустая цепочка обозначается буквой Λ.

Маленькие буквы начала латинского алфавита обозначают символы входного алфавита автомата или терминального алфавита грамматики. Большие латинские буквы обозначают символы вспомогательных алфавитов. Маленькие буквы конца латинского алфавита обозначают цепочки. Эти соглашения обычно позволяют при задании грамматики (автомата) ограничиться выписыванием правил (соответственно команд).

Допускается вольность речи и обозначений, направленная на сокращение текста и принятая без оговорок во многих работах. Так, мы нередко пишем обороты вроде "множество $L \subseteq \Sigma^*$"или "число $m > 1$". Используются распространенные правила умолчания. Например, подразумевается, что термин, вводимый некоторым определением, применяется ко всем объектам, которые указываются данным определением, и только к ним.

Глава 1

НЕКОТОРЫЕ ХАРАКТЕРИЗАЦИИ БЕСКОНТЕКСТНЫХ ЯЗЫКОВ

В данной главе рассматриваются новые характеризации бесконтекстных языков. Каждой из них посвящен отдельный параграф. Порядок параграфов отражает логическую обусловленность одних характеризаций другими; он отвечает и хронологии их появления.

В параграфе 1 определяются понятия Dғрафа и его ядра и доказывается (с помощью понятия магазинного автомата), что Dғрафы действительно характеризуют бесконтекстные языки.

Dғрафы — это представление бесконтекстных языков, которое, с одной стороны, естественным образом обобщает представление регулярных языков диаграммами состояний (подразумеваются состояния конечного автомата), с другой, может рассматриваться как промежуточное звено между магазинными автоматами и морфическими представлениями, введенными Хомским и Шютценберже.

Это очень важные особенности нового формализма. Первая наталкивает на обобщение, а в некоторых случаях (для специальных подклассов бесконтекстных языков) на прямой перенос алгоритмов и схем доказательств, разработанных в теории регулярных языков. Вторая стимулирует осознанное применение ранее практически не работавшей абстракции, другими словами, наталкивает на приложения возможности задать бесконтекстный язык тройкой, которая состоит из двух языков частного (очень простого) вида и морфизма, также имеющего частный вид (в терминах некоторых авторов — "хорошего").

Оба тезиса достаточно выразительно иллюстрируются в настоящей работе. В параграфе 3, например, так называемое праволинейное разложение маршрута D-графа эксплуатирует идею общности между маршрутом протяжения 1 и дугой конечного автомата; КСвыражение, производимое по Dғрафу, получается применением морфизма к КСвыражению, задающему успешные маршруты Dғрафа. Основная теорема параграфа 2 обобщает известную теорему о морфическом представлении регулярных языков и т.д.

Параграф 2 содержит получаемую с помощью понятия Dғрафа новую теорему о морфическом представлении бесконтекстного языка, из которой легко выводится теорема ХомскогоЊШютценберже.

В параграфе 3 рассматривается понятие КСвыражения. Оно появилось в результате исследований по выявлению и адекватному выражению общей сути таких формализмов, как магазинные автоматы, Dғрафы, бесконтекстные грамматики и морфические представления бесконтекстных языков. По существу, это конечное множество цепочек языка, определенным образом размеченных. Разметка указывает места разрастания цепочек и повторяемые при разрастании фрагменты. Одно только сопоставление определения КСвыражения с широко известными очень полезными теоремами о разрастании цепочек бесконтекстного языка позволяет надеяться на полезность нового понятия.

В данной работе понятие КСвыражения послужило для открытия новых характеризаций регулярных языков, повидимому, наименее ограничительных среди известных. Последний результат важен, если, например, считать жизнеспособными технологии вроде представленной в докторской диссертации [Мартыненко 98].

КСвыражения можно считать обобщением регулярных выражений, а теорема об их эквивалентности Dғрафам является аналогом теоремы Клини.

Отметим, что попытки определить аналог регулярных выражений для задания бесконтекстных языков предпринимались давно. Например, Й. Груска [Gruska 71] ввел "характеристику"бесконтекстных языков с использованием объединения, конкатенации и "операционноєимвольной итерации". Он был вынужден использовать "вспомогательные символы"(ср. наш алфавит именованных скобок). Их роль и количество имеют некоторое сходство с таковыми для пар соответствующих именованных скобок. Применил введенную характеристику для определения новых характеристик полулинейных языков и их подклассов.

В определении нашего КСвыражения и его применении явно и систематически используется оригинальное понятие Dязыка. На наш взгляд, понятие КСвыражения является наиболее подходящим воплощением идеи согласованной подстановки, обозначенной еще в 1966–1967 гг.

§1. Dғрафы

Понятие Dғрафа навеяно понятиями конечного и магазинного автоматов и воплощает наше стремление разработать в определенном отношении удачное графическое представление бесконтекстного языка. Оно тесно связано с понятием Dязыка, которое определяется и до некоторой степени изучается в разделе 1.1.

Dязыки обобщают известные языки Дика, точнее, ограниченные языки Дика. Dязыки, подобно языкам Дика, можно интерпретировать как множества скобочных систем, но взаимноөднозначное соответствие открывающих скобок закрывающим теперь не обязательно. Таким образом, Dязык может использовать, например, меньше открывающих скобок, чем закрывающих.

Вводятся названия для некоторых участков скобочных систем и понятия, позволяющие классифицировать скобочные системы по сложности их устройства. Среди этих понятий простейшим (но часто и плодотворно применяемым) является протяжение скобочной системы — наибольшее число цепочек, которые сами являются скобочными системами, и сцепление которых образует рассматриваемую систему. Устанавливается зависимость длины скобочной системы от ее так называемых ширины и глубины.

Вычисления магазинного автомата после его нормализации по способу, представленному в разделе 1.2, становятся фрагментами цепочек определенного D-языка. Это обуславливает полезность введенных в разделе 1.1 понятий для теории магазинных автоматов.

В разделах 1.2–1.3 рассматривается характеризация бесконтекстных языков D-графами. Раздел 1.2 содержит определение Dғрафа и алгоритмы, которые преобразуют магазинный автомат в эквивалентный ему Dғраф и Dғраф в эквивалентный магазинный автомат (языковые описания называют эквивалентными, если они задают один и тот же язык). Термины теории графов и некоторые дополнительные термины, связанные со специфическими весами дуг, позволяют полно и точно отразить подробности работы магазинного автомата. Следует заметить, что отдельные элементы предложенной в разделе 1.2 системы понятий можно найти и в работах других авторов. Нередки также приблизительные аналоги наших понятий и частные их случаи. Вообще, графическое представление машин Тьюринга

и их частных случаев (к которым можно отнести магазинные автоматы) имеет давнюю историю. В настоящее время графические представления магазинных автоматов обсуждаются даже в учебниках; см., например, главу 8 в [Sudkamp 97].

В разделе 1.3 определяются два существенно новых понятия: ядро Dғрафа и развитие — специфическое бинарное отношение на множестве маршрутов, т.е. путей, несущих цепочки заданного Dғрафом языка.

Определение развития апеллирует к понятию цикла. В нем не только охватывается, но и обобщается способ разрастания цепочек бесконтекстного языка, охарактеризованный леммой Огдена.

Ядро — это, по сути, конечное подмножество множества всех маршрутов, рассмотрение которого необходимо и достаточно для решения некоторого вопроса о Dғрафе или соответствующем языке. Точнее, вводится понятие (w,d)ядра, где параметры w и d обеспечивают определенные ограничения на вхождение циклов в рассматриваемые маршруты. Величина параметров может сильно варьироваться от задачи к задаче. Так, в том же разделе доказывается (см. теорему 4), что каждый маршрут "получается развитием"некоторого элемента (2,1)ядра. А в параграфе 3 данной главы доказывается, что, вообще говоря, только (5,5)ядро позволяет ответить на вопрос о так называемой псевдосогласуемости.

Заметим, что в доказательстве леммы 5, помогающей установить справедливость теоремы 4, демонстрируется специфический и весьма употребительный у нас прием доказательства фактов, формулируемых по схеме:

Если некоторый маршрут Dғрафа имеет свойство P, то ядро этого Dғрафа содержит маршрут со свойством P".

Прием состоит в следующем: в произвольном маршруте, обладающем свойством P, фиксируются обуславливающие его участки; в остальных участках удаляются все циклы, при отсутствии которых имеем все еще маршрут. Последний и является элементом ядра, параметры которого определяются числом изначально фиксированных участков и, иногда, некоторой их взаимозависимостью.

1.1. Dязыки. Понятие Dязыка обобщает понятие языка Дика, которое может быть определено, например, следующим образом.

Пусть $\Sigma_($ и $\Sigma_)$ — непересекающиеся алфавиты. Пусть существует биекция

$$\phi : \Sigma_(\to \Sigma_).$$

Тогда язык, порождаемый грамматикой

$$S \to \Lambda \mid aSbS,\ a \in \Sigma_(,\ b = \phi(a),$$

назовем *языком Дика*.

Поведение символов в цепочках языка Дика моделирует поведение скобок. В связи с этим мы будем называть цепочки языков Дика и определяемых далее Dязыков скобочными системами.

Пример 1. К наименьшим по числу видов скобок в скобочных системах относится язык Дика, порождаемый грамматикой

$$S \to \Lambda \mid (S)S.$$

Заметим, что "скобочное поведение"присуще в бесконтекстных языках объектам, которые, на первый взгляд, трудно назвать скобками.

Пример 2. Грамматика
$S \to \Lambda \mid aAbS \mid bBaS$
$A \to \Lambda \mid aAbA$
$B \to \Lambda \mid bBaB$
порождает цепочки, в которых a и b поровну. Она указывает, что минимальный из непустых префиксов такой цепочки, в свою очередь принадлежащих данному языку, является скобочной системой с парой скобок (a, b) или (b, a).

Пример 3. В цепочках, порождаемых грамматикой G с правилами

$$S \to \Lambda \mid aSbS,\ (a, b) \in \{(begin, end), (case, end)\},$$

моделируется поведение символов, ограничивающих составной оператор и оператор варианта в языке ПАСКАЛЬ [Wirth 71]. Отметим, что $L(G)$ не удовлетворяет определению языка Дика.

Пусть $\Sigma_($ и $\Sigma_)$ — непересекающиеся алфавиты. Тогда назовем непустое множество

$$\mathcal{P} \subseteq \Sigma_(\times \Sigma_)$$

Dмножеством , а язык $\mathcal{L}_P$, порождаемый грамматикой

$$S \to \Lambda \mid aSbS,\ (a, b) \in \mathcal{P},$$

Dязыком (*над Dмножеством* $\mathcal{P}$).
Язык Дика является Dязыком. Обратное неверно, как показывает пример 3.

Пусть $\Delta = \Sigma_(\cup \Sigma_)$, $w \in \Delta^*$. Пусть отображение

$$\mu : \Delta^* \to \Delta^*$$

определяется равенством $\mu(w) = t$, где t есть результат применения к w следующих действий:

$t := w$;
пока для некоторых $(a, b) \in \mathcal{P}$ и $w_1, w_2 \in \Delta^*$ справедливо равенство $t = w_1abw_2$, выполнять присваивание $t := w_1w_2$.

Тогда назовем μ D-*отображением.*
Заметим, что Dотображение стирает цепочки Dязыка, и

$$\mathcal{L}_P = \{w \in \Delta^* \mid \mu(w) = \Lambda\}.$$

Пусть цепочка $y \in \Delta^*$ такова, что для некоторых x и z из Δ^* цепочка $xyz \in \mathcal{L}_P$. Тогда назовем y *фрагментом* (скобочной системы xyz).

Пусть $x \in \mathcal{L}_P$. Определим рекурсивно неотрицательное целое $parti(x)$ и назовем его *протяжением* цепочки x:
1) $parti(\Lambda) = 0$;
2) если $y \in \mathcal{L}_P$, $(a, b) \in \mathcal{P}$ и $x = ayb$, то $parti(x) = 1$;
3) если $y, z \in \mathcal{L}_P$ и $x = yz$, то $parti(x) = parti(y) + parti(z)$.

Назовем *шириной* цепочки $x \in \mathcal{L}_P$ число $width(x) = max\{parti(z) |\ z \in \mathcal{L}_P$, $x = uzy$ для некоторых фрагментов u и $y\}$.

Назовем *глубиной* цепочки $x \in \mathcal{L}_P$ число $depth(x)$, рекурсивно определяемое следующим образом:

1) $depth(\Lambda) = 0$;

2) если $x, y \in \mathcal{L}_P$ и $(a, b) \in \mathcal{P}$, то $depth(axb) = depth(x) + 1$, $depth(xy) = max(depth(x), depth(y))$.

Пример 4. Пусть $\mathcal{P} = \{(a, b),\ (a, c)\}$, $x = aabacaccac$. Тогда $x \in \mathcal{L}_P$, $parti(x) = depth(x) = 2$, $width(x) = 3$. Действительно, цепочка x является сцеплением двух непустых скобочных систем, в которых начальный и конечный символы парны друг другу: $aabacacc$ и ac. Первая из этих систем имеет глубину 2, вторая — 1. Фрагмент $abacac$ имеет протяжение 3.

Следующая теорема дает верхнюю оценку длин скобочных систем с ограниченными глубиной и шириной.

Теорема 1. Пусть w, d — неотрицательные целые. Пусть цепочка $\psi \in \mathcal{L}_P$ удовлетворяет неравенствам $width(\psi) \leq w$, $depth(\psi) \leq d$. Тогда $|\psi| \leq g_{w,d}$, где число $g_{w,d}$ определяется следующими рекуррентными соотношениями:

1) $g_{w,1} = 2w$,

2) $g_{w,d} = (g_{w,d-1} + 2)w, \quad d > 1$.

Доказательство. Пусть $n = parti(\psi)$. Тогда цепочка ψ представляется в виде $\psi = \psi_1 \dots \psi_n$, где $\psi_i \in \mathcal{L}_P - \{\Lambda\}$ для $i = 1, \dots, n$. Заметим, что $n \leq w$. Очевидно,

$$| \psi | = \sum_{i=1}^{n} | \psi_i | .$$

Докажем теорему индукцией по d. При $d = 1$ имеем $| \psi_i | = 2$ для $1 \leq i \leq n$. Следовательно,

$$| \psi | = 2n \leq 2w = g_{w,1}.$$

Пусть $d > 1$. Предположим, что теорема верна для цепочек, глубина которых не превосходит $d-1$, а ширина не превосходит w. Из выбора числа n следует, что цепочка ψ_i, $1 \leq i \leq n$, имеет вид $a_i \xi_i b_i$ для некоторых $\xi_i \in \mathcal{L}_P$, $(a_i, b_i) \in \mathcal{P}$, причем $depth(\xi_i) \leq d - 1$. Из определения ширины следует, что $width\ (\xi) \leq width(\psi)$ для любого фрагмента $\xi \in \mathcal{L}_P$ цепочки $\psi \in \mathcal{L}_P$. Следовательно, $width(\xi_i) \leq w$, так как ξ_i — фрагмент цепочки ψ.

По предположению индукции $| \xi_i | \leq g_{w,d-1}$. Следовательно,

$$|\psi_i| = |\xi_i| + |a_i b_i| \leq g_{w,d-1} + 2,$$

а

$$| \psi | \leq (g_{w,d-1} + 2)n \leq (g_{w,d-1} + 2)w = g_{w,d}$$

□

1.2. Dғрафы и магазинные автоматы

1.2.1. Определение Dғрафа. Пусть для любого Dмножества $\mathcal{P} \subseteq \Sigma_(\times \Sigma_)$ записи $Left(\mathcal{P})$ и $Right(\mathcal{P})$ обозначают соответственно множества

$$\{a \in \Sigma_(| \exists b \in \Sigma_) \quad (a, b) \in \mathcal{P}\}$$

и

$$\{b \in \Sigma_) | \exists a \in \Sigma_(\quad (a, b) \in \mathcal{P}\}.$$

Назовем *Dғрафом* шестерку $D = (\Sigma,\ V,\ \mathcal{P},\ \lambda,\ P_0,\ F)$, где:
Σ — *алфавит пометок дуг*;
V — конечное множество *вершин*;
$P_0 \in V$ — *входная вершина*;
$F \subseteq V$ — *множество заключительных вершин*;
$\mathcal{P}$ — Dмножество; объединение

$$E(D) = Left(\mathcal{P}) \cup Right(\mathcal{P})$$

есть множество *дуг*; это ориентированные нагруженные дуги;
$\lambda : E(D) \to V \times (\Sigma \cup \{\Lambda\}) \times V$ — *функция положения* (дуги в Dғрафе); элементы тройки

$$(P, a, Q) \in V \times (\Sigma \cup \{\Lambda\}) \times V,$$

которая удовлетворяет равенству $(P, a, Q) = \lambda(\pi)$ для некоторой дуги π, интерпретируются соответственно как начальная вершина, пометка и конечная вершина этой дуги.

Прежде чем дать определение задаваемого Dғрафом языка и привести пример Dғрафа, введем несколько обозначений и терминов, полезных как для этих ближайших целей, так и в дальнейшем.

Пусть для любого пути T $beg(T)$ и $end(T)$ обозначают соответственно его начальную и конечную вершины. Пусть $pair(T)$ обозначает упорядоченную пару $(beg(T), end(T))$.

Понятие *пометки пути* в графе D определим рекурсивно: пометка пустого (или тривиального) пути пуста, а для пути $T\pi$, в котором $\pi \in E(D)$ и, следовательно, $\lambda(\pi) = (P, a, Q)$ для некоторых $P, Q \in V$ и $a \in \Sigma \cup \{\Lambda\}$, пометка есть $\omega(T\pi) = \omega(T)a$.

Элемент множества путей

$$Sentences(D) = \{T \in \mathcal{L}_P | beg(T) = P_0,\ end(T) \in F\}$$

назовем *предложением* Dғрафа D.

Будем говорить, что Dғраф D *определяет язык*

$$L(D) = \{\omega(T) \mid T \in Sentences(D)\}.$$

Условимся опускать разделители в записи тройки $\lambda(\pi)$, если это не мешает различить ее элементы.

Пример 5. Рассмотрим Dғраф

$$D_1 = (\{a, b\},\ \{P, Q\},\ \{(1, 2),\ (1, 3)\},\ \lambda_1,\ P,\ \{Q\}),$$

где $\lambda_1(1) = PaP$, $\lambda_1(2) = PbQ$, $\lambda_1(3) = QbQ$.

Множество всех его путей из входной вершины P в заключительную вершину Q можно описать формулой

$$\{1^k 23^l | k, l \geq 0\}.$$

$$Sentences(D_1) = \{1^n 23^{n-1} | n \geq 1\}.$$

$$L(D_1) = \{a^n b^n | n \geq 1\}.$$

Теперь наша цель — доказать следующую теорему.

Теорема 2. Язык тогда и только тогда является бесконтекстным, когда он определяется некоторым Dғрафом $\square$

Для этого сначала конструктивно докажем, что произвольному магазинному автомату эквивалентен некоторый специфический Dғраф, затем используем построение из данного доказательства для преобразования произвольного Dғрафа в эквивалентный магазинный автомат.

1.2.2. Преобразование магазинного автомата в Dғраф

Магазинный автомат есть семерка $M = (K, \Sigma, \Gamma, Z_0, \delta, p_0, F)$, где:
K — конечное множество *состояний*;
Σ — *входной алфавит*;
Γ — *магазинный алфавит*;
$Z_0 \in \Gamma$ — маркер дна магазина;
$p_0 \in K$ — *начальное состояние*;
$F \subseteq K$ — множество *заключительных состояний*;
$\delta \subseteq K \times (\Sigma \cup \{\Lambda\}) \times \Gamma \times K \times \Gamma^*$ — конечное множество *команд*.

Это определение отличается от приведенного в [Ginsburg 66] только интерпретацией элемента δ. Понятие команды удобно для сопоставления магазинных автоматов с Dғрафами.

Язык $L(M)$, *допускаемый автоматом* M, определяется с помощью понятия конфигурации и бинарного отношения *достижимости*, заданного на множестве конфигураций.

Далее, если не оговорено иное, считаем, что

$$p, q \in K, \quad a \in \Sigma \cup \{\Lambda\}, \quad X, Z \in \Gamma, \quad \gamma \in \Gamma^*.$$

Правый символ магазинной цепочки отвечает верху магазина. Команду обозначаем записью

$$(p, a, Z) \to (q, \gamma).$$

Пусть

$$(p, ax, \gamma_1 Z), (q, x, \gamma_1\gamma) \in K \times \Sigma^* \times \Gamma^*,$$
$$\theta = (p, a, Z) \to (q, \gamma) \in \delta.$$

Тогда пара

$$((p, ax, \gamma_1 Z), (q, x, \gamma_1\gamma))$$

принадлежит бинарному отношению $\overset{\theta}{\models}$. Определим отношение $\models_M$ как объединение $\cup_{\theta\in\delta} \overset{\theta}{\models}$.

Пусть $p \in K$, $x, y \in \Sigma^*$, $\gamma \in \Gamma^*$ и

$$((p_0, x, Z_0), (p, y, \gamma)) \in \models_M^* .$$

Тогда назовем тройку (p, y, γ) *конфигурацией магазинного автомата* M. Конфигурацию (p_0, x, Z_0) назовем *начальной*, конфигурацию (p, Λ, γ), где $p \in F$, — *заключительной*.

Теперь язык $L(M)$ можно определить формулой

$$L(M) = \{x \in \Sigma^* | \exists (p \in F, \gamma \in \Gamma^*)\ (p_0, x, Z_0) \models_M^* (p, \Lambda, \gamma)\}.$$

Преобразуем магазинный автомат так, чтобы задача построения эквивалентного ему D-графа предельно упростилась.

Пусть $\theta = (p, a, Z) \to (q, \gamma) \in \delta$, $\gamma = W_1 \dots W_n$, $n \geq 0$, $W_i \in \Gamma$ для $1 \leq i \leq n$, $+, - \notin \Gamma$. Назовем *следом команды* θ цепочку $\alpha(\theta)$, где $\alpha(\theta) = -Z + W_1 \dots + W_n$ при $n \geq 1$ и $Z \neq W_1$, $\alpha(\theta) = +W_2 \dots + W_n$ при $n > 1$ и $Z = W_1$, $\alpha(\theta) = \Lambda$ при $\gamma = Z$, $\alpha(\theta) = -Z$ при $\gamma = \Lambda$. Здесь минус символизирует удаление из магазина символа Z, плюс — запись в магазин символа цепочки γ.

Пусть магазинный автомат $M = (K, \Sigma, \Gamma, Z_0, \delta, p_0, F)$ удовлетворяет следующим условиям:

1) след любой команды есть $+X$ или $-X$ для некоторого $X \in \Gamma$;

2) любая конфигурация имеет вид $(p, x, Z_0\gamma)$, где $\gamma \in (\Gamma - \{Z_0\})^*$, $p \in K$, $x \in \Sigma^*$;

3) заключительная конфигурация имеет вид (f, Λ, Z_0), где $f \in F$. Тогда назовем автомат M *совершенным*.

Следующая лемма означает, что, ограничиваясь рассмотрением совершенных магазинных автоматов, мы не ограничиваем класса бесконтекстных языков.

Лемма 1. Для любого магазинного автомата M существует совершенный магазинный автомат M', такой, что $L(M') = L(M)$.

Доказательство. Лемма верна, так как следующий алгоритм 1 строит по любому магазинному автомату M эквивалентный совершенный магазинный автомат M'. Алгоритм вводит дополнительные состояния, магазинные символы и команды, чтобы избавиться от команд, выполняющих с магазином операцию, недозволенную в случае совершенного автомата □

Команду θ назовем *нейтральной*, если $\alpha(\theta) = \Lambda$, *накапливающей*, если $\alpha(\theta) = +X$ для некоторого $X \in \Gamma$, и *стирающей*, если $\alpha(\theta) = -Z$.

Алгоритм 1. Совершенствование магазинного автомата.

Вход. Магазинный автомат $M' = (K', \Sigma, \Gamma', Z_0', \delta', p_0, F')$.

Выход. Магазинный автомат $M = (K, \Sigma, \Gamma, Z_0, \delta, p_0, F)$.

Шаг 1. (Обеспечивает вид $Z_0\gamma$, где $\gamma \in (\Gamma - \{Z_0\})^*$, третьего элемента конфигурации.)

$\Gamma := \Gamma' \cup \{Z_0\}$, $Z_0 \notin \Gamma'$;

$\delta := \delta' \cup \{(p_0, \Lambda, Z_0) \to (p_0, Z_0 Z_0')\}$.

Шаг 2. (Обеспечивает равенство $\gamma = Z_0$ для третьего элемента γ заключительной конфигурации.)

$K := K' \cup \{f_1, f\}$, где $f_1, f \notin K'$;

$\delta := \delta \cup \{(p, \Lambda, Z) \to (f_1, Z) \mid p \in F',\ Z \in \Gamma\} \cup$

$\{(f_1, \Lambda, Z) \to (f_1, \Lambda) \mid Z \in \Gamma'\} \cup$

$\{(f_1, \Lambda, Z_0) \to (f, Z_0)\}$;

$F := \{f\}$.

Шаг 3. (Преобразует команды так, что каждая команда результирующего автомата является либо накапливающей, либо стирающей.)

$K := K \cup \{p_\theta \notin K \mid \theta = (p, a, Z) \to (q, W_1 \dots W_n),\ Z \neq W_1,\ n \geq 1\}$.

Каждую команду вида

$$\theta = (p, a, Z) \to (q, W_1 \dots W_n),$$

где $Z \neq W_1$ и $n \geq 1$, заменить командами

$$(p, a, Z) \to (p_\theta, \Lambda),\ (p_\theta, \Lambda, X) \to (q, XW_1 \dots W_n),\ X \in \Gamma.$$

$K := K \cup \{p_{\theta,i} \notin K \mid \theta = (p, a, W_1) \to (q, W_1 \dots W_n),\ n > 2,\ 2 \leq i \leq n-1\}$.

Каждую команду вида

$$\theta = (p, a, W_1) \to (q, W_1 ... W_n),$$

где $n > 2$, заменить командами

$(p, a, W_1) \to (p_{\theta,2}, W_1 W_2)$,

$(p_{\theta,i}, \Lambda, W_i) \to (p_{\theta,i+1}, W_i W_{i+1})$, $2 \leq i \leq n-2$,

$(p_{\theta,n-1}, \Lambda, W_{n-1}) \to (q, W_{n-1} W_n)$.

$\Gamma := \Gamma \cup \{Z_\theta \notin \Gamma \mid$ команда $\theta \in \delta$ является нейтральной$\}$.

Заменить любую нейтральную команду

$$\theta = (p, a, Z) \to (q, Z)$$

двумя командами

$$(p, a, Z) \to (p, ZZ_\theta),\ (p, \Lambda, Z_\theta) \to (q, \Lambda).$$

Далее считаем магазинный автомат

$$M = (K, \Sigma, \Gamma, Z_0, \delta, p_0, F)$$

совершенным. Определим для него понятие вычисления, имеющее вполне традиционный смысл. Специфика совершенного автомата позволит дать вычислениям естественную графическую трактовку.

Пусть $\gamma = X_1 \dots X_n$, где $n \geq 0$ и $X_i \in \Gamma$ для $0 \leq i \leq n$. Тогда будем обозначать цепочку $+X_1 \dots + X_n$ записью $+\gamma$, а цепочку $-X_n \dots - X_1$ записью $-\gamma$.

Пусть $(p, x, \gamma'\gamma)$ — конфигурация, $|\gamma| \geq 1$. Тогда назовем пару $(p, +\gamma)$ *памятью*.

Заметим, что при

$$\Sigma_(= \{+X \mid X \in \Gamma\}$$

и

$$\Sigma_) = \{-X \mid X \in \Gamma\}$$

множество $\Delta = \Sigma_(\cup \Sigma_)$ можно интерпретировать как алфавит языка Дика. Следовательно, законно применение Dѳотображения к цепочкам из Δ^*.

Пусть ρ — память. Рекурсивно определим *вычисление* T (*над памятью* ρ), его *пометку* $\omega(T)$, *след* $\alpha(T)$, *конечную память* $ecol(T)$ и *длину* $|T|$:

1) пара $T = (\rho, \Lambda)$ с пустым вторым элементом есть (пустое) вычисление; $\omega(T) = \alpha(T) = \Lambda$, $ecol(T) = \rho$, $|T| = 0$;

2) пусть $T_1 = (\rho, \Theta)$ — вычисление и $ecol(T_1) = (p, z + Z)$, где $Z \in \Gamma$, $z \in \Delta^*$; пусть $\theta = (p, a, Z) \to (q, \gamma)$ — такая команда, что

$$w = \mu(z + Z\alpha(\theta)) \in \Sigma_(^+.$$

Тогда пара $T = (\rho, \Theta\theta)$ есть вычисление, причем $\omega(T) = \omega(T_1)a$, $\alpha(T) = \alpha(T_1)\alpha(\theta)$, $ecol(T) = (q, w)$, $\mid T \mid = \mid T_1 \mid +1$.

Назовем память $bcol(T) = \rho$ *начальной памятью вычисления* T. Назовем *зарядом вычисления* T цепочку $\mu(T) = \mu(\alpha(T))$.

Пусть $p \in K$, $W \in \Gamma$, γ_1, $\gamma_2 \in \Gamma^*$, $\rho = (p, +W\gamma_1)$ — память, $T = (\rho, \Theta)$ — вычисление над памятью ρ, $\mu(T) = -\gamma_1 + \gamma_2$. Тогда назовем T *маршрутом* (*над памятью* ρ); если $\mid T \mid = 1$, то будем также называть T *дугой*.

Заметим, что если маршрут T пуст, то

$$ecol(T) = \rho = (p, +W).$$

Если T не пуст и последним элементом в последовательности Θ его команд является команда, переводящая автомат в состояние q, то

$$ecol(T) = (q, +W\gamma_2).$$

Пусть $p, q \in K$, $w \in \Delta^*$, $W \in \Gamma$, γ_1, $\gamma_2 \in \Gamma^*$, $\Theta \in \delta^*$ таковы, что

$$T = ((p, w + W\gamma_1), \Theta)$$

является вычислением и $\mu(T) = -\gamma_1 + \gamma_2$. Тогда назовем *основой вычисления* T маршрут

$$\beta(T) = ((p, +W\gamma_1), \Theta).$$

Определим на множестве вычислений отношение $\sim_M$ ($\sim$, если автомат подразумевается) следующим образом:

$$(T_1, T_2) \in \sim \Longleftrightarrow \beta(T_1) = \beta(T_2).$$

Введенное отношение является отношением эквивалентности; эквивалентные вычисления имеют одну и ту же основу. В дальнейших рассмотрениях будут участвовать только маршруты, представляющие классы эквивалентности, на которые множество вычислений разбивается отношением $\sim$. Это оправдано тем соображением, что внутренние свойства вычисления T не зависят от фрагмента, "отсекаемого"для получения основы $\beta(T)$ этого вычисления.

Пусть $(p, +Z)$ — начальная память некоторого пустого маршрута. Тогда пару (p, Z) назовем *вершиной автомата* (и упомянутого маршрута). Если p — заключительное состояние, то назовем вершину (p, Z) *заключительной*. Пару (p_0, Z_0) назовем *входной вершиной*.

Отметим, что в случае совершенного магазинного автомата второй элемент заключительной вершины всегда есть Z_0.

Лемма 2. 1. Пустой маршрут может быть представлен своей вершиной. 2. Непустой маршрут $(\rho, \theta_1 \dots \theta_m)$, где $m \geq 1$ и $\theta_i \in \delta$ для $1 \leq i \leq m$, может быть представлен последовательностью $\pi_1 \dots \pi_m$ дуг, вычисляемых следующим образом:

$$\pi_1 = \beta((\rho, \theta_1)),$$

$$\pi_i = \beta((ecol((\rho, \theta_1 \dots \theta_{i-1})), \theta_i))$$

для $i = 2, \dots, m$.

Доказательство. Часть 1 леммы следует из определений пустого вычисления, маршрута и вершины. Часть 2 нетрудно доказать индукцией по числу m команд во втором из элементов маршрута $\square$

Пусть маршрут

$$T = T_0\pi_1T_1T_2T_3\pi_3T_4$$

таков, что π_1 и π_3 являются дугами и

$$\mu(T_2) = \mu(T_1T_2T_3) = \mu(\pi_1T_1T_2T_3\pi_3) = \Lambda.$$

Тогда назовем тройку

$$(\pi_1T_1,\ T_2,\ T_3\pi_3)$$

гнездом (маршрута T). Иногда будем называть гнездом и маршрут $\pi_1T_1T_2T_3\pi_3$. Будем говорить, что участки π_1T_1, $T_3\pi_3$ *парны* (друг другу), или *образуют гнездо* в маршруте T.

Пусть маршрут T таков, что $bcol(T) = (p_0, +Z_0)$, $ecol(T) = (f, +Z_0)$ для $f \in F$. Тогда назовем T *успешным маршрутом* (или *предложением*). Из определений следует, что совокупность пометок успешных маршрутов совершенного магазинного автомата есть язык, допускаемый автоматом.

Пусть $V(M)$ есть множество всех вершин, определяемых пустыми участками успешных маршрутов автомата M, $\mathcal{P}(M)$ — множество всех пар дуг (π_1, π_2), таких, что π_1 и π_2 парны в некотором успешном маршруте автомата M. Тогда ясно, что Dграф

$$Graph(M) = (\Sigma,\ V(M),\ \mathcal{P}(M),\ \lambda(M),\ (p_0, Z_0),\ \{(p, Z_0)|p \in F\}),$$

где функция положения $\lambda(M)$ сопоставляет дуге

$$\pi \in Left(\mathcal{P}(M)) \cup Right(\mathcal{P}(M))$$

тройку $(beg(\pi), \omega(\pi), end(\pi))$, эквивалентен автомату M. Dграф $Graph(M)$ будем называть *графом магазинного автомата* M. $\mathcal{P}(M)$ будем называть Dмножеством автомата M.

Конечность множеств $V(M)$ и $\mathcal{P}(M)$ с очевидностью следует из конечности множеств K, Σ, Γ и δ. Один из способов доказательства возможности построения множеств $V(M)$ и $\mathcal{P}(M)$ связан с понятием ядра, которое обсуждается в разделе 1.3. Эта возможность выводится из леммы 6 раздела 1.3. Таким образом, построение Dграфа по магазинному автомату завершено.

Пример 6. Пусть

$$M = (\{p_0, p_1, p_2, f\}, \{a, b\}, \{a, Z_0\}, Z_0, p_0, \delta, \{f\}),$$

где множество δ задается следующим перечнем команд:
$(p_0, a, Z) \to (p_0, Za),\ Z \in \{Z_0, a\}$,
$(p, b, a) \to (p_1, \Lambda),\ p \in \{p_0, p_1\}$,
$(p_1, \Lambda, Z_0) \to (p_2, Z_0a)$,
$(p_2, \Lambda, a) \to (f, \Lambda)$,
$(f, b, Z_0) \to (p_2, Z_0a)$.
Тогда M есть детерминированный совершенный магазинный автомат. Он допускает нерегулярный язык

$$\{a^mb^n|m \geq 1,\ n \geq m\}.$$

Вершинами Dграфа $Graph(M)$ являются пары

$$(p_0, Z_0),\ (p_0, a),\ (p_1, a),\ (p_1, Z_0),\ (p_2, a),\ (f, Z_0).$$

Здесь (p_0, Z_0) — входная вершина, (f, Z_0) — заключительная.

$$\mathcal{P}(Graph(M)) = \mathcal{P}(M) = \{(1,5),(1,6),(2,3),(2,4),(7,8),(9,8)\},$$

где

$\lambda(1) = (p_0, Z_0)a(p_0, a)$,
$\lambda(2) = (p_0, a)a(p_0, a)$,
$\lambda(3) = (p_0, a)b(p_1, a)$,
$\lambda(4) = (p_1, a)b(p_1, a)$,
$\lambda(5) = (p_0, a)b(p_1, Z_0)$,
$\lambda(6) = (p_1, a)b(p_1, Z_0)$,
$\lambda(7) = (p_1, Z_0)\Lambda(p_2,\ a)$,
$\lambda(8) = (p_2, a)\Lambda(f, Z_0)$,
$\lambda(9) = (f, Z_0)b(p_2, a)$.

Дуги 3 и 5 отвечают стирающей команде

$$(p_0, b, a) \to (p_1, \Lambda).$$

Их конечные вершины демонстрируют все те символы (и только те), которые бывают на верхушке магазина после выполнения данной команды. Аналогично, дуги 4 и 6 отвечают команде

$$(p_1, b, a) \to (p_1, \Lambda).$$

В то же время стирающей команде

$$(p_2, \Lambda, a) \to (f, \Lambda)$$

отвечает лишь одна дуга.

Множество успешных маршрутов автомата M задается формулой

$$\{12^k234^k678(98)^l | k, l \geq 0\} \cup \{1578(98)^l | l \geq 0\}.$$

Заметим, что путь (25) не является маршрутом. В самом деле, начальная и конечная памяти дуги 5 определяются однозначно:

$$bcol(5) = (p_0, +Z_0 + a),\ ecol(5) = (p_1, +Z_0).$$

А множество конечных памятей маршрутов, которые имеют начальную память $(p_0, +Z_0)$ и заканчиваются дугой 2, есть

$$\{(p_0, +Z_0(+a)^k | k \geq 2\}.$$

Оно не содержит памяти $bcol(5)$.

1.2.3. Преобразование Dғрафа в магазинный автомат. Применим понятие графа магазинного автомата для доказательства следующего факта: для каждого Dғрафа существует эквивалентный магазинный автомат. Построим алгоритм, который по произвольному Dғрафу D сначала получает эквивалентный Dғраф $G(D)$, являющийся графом некоторого магазинного автомата, затем преобразует дуги полученного Dғрафа в команды автомата.

Алгоритм 2. Построение магазинного автомата по Dғрафу

Вход. Dғраф $D = (\Sigma,\ V,\ \mathcal{P},\ \lambda,\ P_0,\ F)$.

Вспомогательные обозначения. $Z_0 \notin \mathcal{P}$; $G(D) = (\Sigma,\ VPDA,\ \mathcal{P}PDA,\ \lambda PDA,\ (P_0, Z_0),\ \{(P, Z_0) | P \in F\})$ — Dғраф, множества $VPDA$, $\mathcal{P}PDA$ и функция λPDA которого определяются шагами алгоритма; $f \notin V$.

Выход. Магазинный автомат $M(D)$, определяемый шагом 3 алгоритма.

Шаг 1. $\mathcal{P}PDA := \emptyset;\ \ VPDA := \{[P_0, Z_0]\}$.

Шаг 2. Пока множество $VPDA$ пополняется новыми вершинами, выполнять следующие два действия:

1) положить новое значение множества $\mathcal{P}PDA$ равным объединению $\mathcal{P}PDA$ и множества

$$\{([P,Z]\omega(\pi_1)[end(\pi_1),\pi_1],\ [beg(\pi_2),\pi_1]\omega(\pi_2)[end(\pi_2,Z])|$$
$$[P,Z] \in VPDA;\ (\pi_1,\pi_2) \in \mathcal{P};\ beg(\pi_1) = P\};$$

2) добавить в $VPDA$ вершины дуг, образующих пары из $\mathcal{P}PDA$.

Шаг 3. Положить $M(D)$ равным семерке $(V \cup \{f\}, \Sigma, Left(\mathcal{P}) \cup \{Z_0\}, Z_0, \{(p_1, a_1, Z) \to (q_1, ZX), (p_2, a_2, X) \to (q_2, \Lambda)|\ ([p_1, Z]a_1[q_1, X],\ [p_2, X]a_2[q_2, Z]) \in \mathcal{P}PDA\} \cup \{(P, \Lambda, Z_0) \to (f, Z_0)|P \in F\}, P_0, \{f\})$.

Положить $\lambda PDA(PaQ)$ равным (P, a, Q) для $P, Q \in VPDA$, $a \in \Sigma \cup \{\Lambda\}$, $PaQ \in Left(\mathcal{P}PDA) \cup Right(\mathcal{P}PDA)$.

Термин "маршрут", введенный при построении графа магазинного автомата, будем применять и в случае произвольного Dграфа $(\Sigma, V, \mathcal{P}, \lambda, P_0, F)$, т.е. будем использовать слово "маршрут"как сокращение оборота "путь, который является фрагментом некоторой цепочки Dязыка $\mathcal{L}_P$". Будем называть *нейтральным* маршрут, который является цепочкой Dязыка $\mathcal{L}_P$. Это уменьшит насыщенность текста формулами. Слово "нейтральный"напоминает о "заряженности"маршрутов магазинного автомата. Нейтральный маршрут магазинного автомата имеет пустой заряд и является основой вычислений, которые не стирают никакую часть исходного содержимого магазина, но обязательно стирают собственные записи в магазине, т.е., в конце концов, ничего к нему не добавляют.

Нетрудно заметить, что маршруты графа $G(D)$ отличаются от маршрутов графа D только тем, что каждая их вершина содержит дополнительно некоторый символ магазинного алфавита автомата $M(D)$. Символ, отвечающий конечной вершине маршрута T графа D, где $beg(T) = P_0$, можно охарактеризовать следующим образом: если T нейтрален, это Z_0, иначе это последняя из его дуг, для которых T не содержит парной дуги.

Утверждение 1. $L(M(D)) = L(G(D)) = L(D)$.

Доказательство. Естественно считать, что запись T_1PT_2, где

$$end(T_1) = P = beg(T_2),$$

обозначает тот же путь, что и запись T_1T_2. Используем это соглашение уже в определении следующей функции, помогающей описать соответствие предложений графов D и $G(D)$.

Функция $\Psi(T, Z)$, где T — нейтральный маршрут графа D, $Z \in \{Z_0\} \cup Left(\mathcal{P})$, определяется следующими рекуррентными соотношениями:

1) если маршрут T пуст, то

$$\Psi(T, Z) = [P, Z],$$

где P — вершина маршрута T;

2) если $T = \pi_1 T_1 \pi_2 T_2$, где T_1 и T_2 нейтральны, $(\pi_1, \pi_2) \in \mathcal{P}$, то

$$\Psi(T, Z) = \psi_1 \Psi(T_1, \pi_1) \psi_2 \Psi(T_2, Z),$$

где

$$\psi_1 = [beg(\pi_1), Z]\omega(\pi_1)[end(\pi_1), \pi_1],$$
$$\psi_2 = [beg(\pi_2), \pi_1]\omega(\pi_2)[end(\pi_2), Z].$$

Сопоставляя данное определение с алгоритмом 2, видим, что значение функции имеет форму маршрута графа $G(D)$, но, возможно, таковым не является, так как пары

$$[beg(\pi_1), Z], \quad [end(\pi_2), Z]$$

не обязаны входить в $VPDA$, а образование

$$([beg(\pi_1), Z]\omega(\pi_1)[end(\pi_1), \pi_1], [beg(\pi_2), \pi_1]\omega(\pi_2)[end(\pi_2), Z])$$

не обязано входить в $\mathcal{P}PDA$.

Пусть $ExtP$ обозначает следующее расширение множества $\mathcal{P}PDA$:

$$ExtP = \{([p_1, Z]a_1Q_1, Q_2a_2[p_2, Z])|\exists X \in \{Z_0\} \cup Left(\mathcal{P})$$
$$([p_1, X]a_1Q_1, Q_2a_2[p_2, X]) \in \mathcal{P}PDA\}.$$

Тогда ясно, что значение функции однозначно определяет некоторую цепочку x Dязыка $\mathcal{L}_{ExtP}$, который содержит в себе Dязык $\mathcal{L}_{\mathcal{P}PDA}$ и, следовательно, его подмножество $Sentences(G(D))$. Существенно, что цепочка x либо является нейтральным маршрутом графа $G(D))$, либо отличается от такового лишь магазинным символом в конечных вершинах его нейтральных начальных участков.

Легко доказать индукцией по $k \geq 0$, что началу цепочки $\Psi(T, Z)$, которое имеет длину $2k$ и принадлежит Dязыку $\mathcal{L}_{ExtP}$, отвечает конечная "вершина"

$$[end(T), Z] \in V \times (\{Z_0\} \cup Left(\mathcal{P})).$$

Учитывая конструкцию Dграфа $G(D))$, получаем отсюда равенство

$$Sentences(G(D)) = \{\Psi(T, Z_0)|T \in Sentences(D)\}.$$

Значит, $L(G(D)) = L(D)$. Следующее равенство вытекает из построения автомата $M(D)$: $Sentences(Graph(M(D))) = \{T_1T_2|T_1 \in Sentences(G(D))$; T_2 отвечает команде с пустым следом, переводящей автомат $M(D)$ в заключительное состояние f; $\omega(T_2) = \Lambda\}$. Следовательно, $L(M(D)) = L(D)$ $\square$

Пример 7. Построим с помощью алгоритма 2 магазинный автомат $M(D_1)$, эквивалентный Dграфу D_1, определенному в примере 5. Выпишем его команды в порядке построения определяющих их дуг графа $G(D_1)$:
$(P, a, Z_0) \to (P, Z_0\ PaP)$,
$(P, b, PaP) \to (Q, \Lambda)$,
$(Q, b, PaP) \to (Q, \Lambda)$,
$(P, a, PaP) \to (P, PaP\ PaP)$,
$(Q, \Lambda, Z_0) \to (f, Z_0)$.
Данный пример показывает необходимость особого заключительного состояния $f \notin V$ (см. алгоритм 2) и команд вида

$$(Q, \Lambda, Z_0) \to (f, Z_0),$$

не отвечающих никакой паре из $\mathcal{P}PDA$. При попытке сделать заключительным какоелибо из состояний, служивших вершинами исходного Dграфа, вместо $M(D_1)$ получается автомат, допускающий префиксы цепочек рассматриваемого языка, ему не принадлежащие.

Заметим, что в статьях и монографиях, в которых строится пример магазинного автомата для языка $\{a^n b^n | n \geq 1\}$, используется именно такой принцип организации автомата, какой наблюдается в случае автомата $M(D_1)$. Автомат является детерминированным и работает в реальном времени, пока цепочка языка не прочитана. Первая половина входной цепочки целиком поступает в магазин в той или иной кодировке.

1.3. Ядро Dғрафа

1.3.1. Определение ядра. В определении ядра Dғрафа большую роль играет следующее понятие цикла, несколько суженное по сравнению с употребляемым в теории графов.

Пусть T — непустой маршрут и $beg(T) = end(T)$. Тогда назовем маршрут T *циклом*.

Таким образом, циклом здесь будет называться только такой циклический путь, который является маршрутом.

Понятия гнезда и парных участков, определенные в терминах магазинных автоматов, будем употреблять для произвольных Dғрафов. Ясно, что для переноса этих определений на общий случай достаточно заменить требование пустоты зарядов некоторых маршрутов на требование нейтральности этих маршрутов.

Пусть T — нейтральный маршрут Dғрафа D, w и d — положительные целые. Пусть T удовлетворяет следующим условиям: если $T_1, \ldots, T_m$ — такие нейтральные циклы, что цикл $T_1 \ldots T_m$ является участком маршрута T, то $m \leq w$; если $T_{11}, \ldots, T_{1m}, T_{21}, T_{31}, \ldots, T_{3m}$ таковы, что $T_{2(i+1)} = T_{1i} T_{2i} T_{3i}$, $1 \leq i \leq m$, есть гнездо маршрута T, образованное циклами T_{1i} и T_{3i}, то $m \leq d$. Тогда назовем T (w, d)-*каноном* Dғрафа D.

Множество (w, d)канонов графа D назовем (w, d)-*ядром* графа и обозначим через $Core(D, w, d)$. Ради краткости пишем далее "канон", "ядро"и "$Core(D)$"вместо "(w, d)канон", " (w, d)ядро"и " $Core(D, w, d)$"соответственно.

Так как нейтральный маршрут является цепочкой Dязыка над алфавитом дуг, можно говорить о ширине и глубине нейтрального маршрута. Для получения верхних оценок ширины и глубины канона полезно

Утверждение 2. Пусть k, m и n — натуральные числа и

$$k > (n-1)m.$$

Кортеж k натуральных чисел, не превосходящих m, содержит не менее n одинаковых чисел.

Доказательство. Пусть число $1 \leq i \leq m$ входит в рассматриваемую последовательность $k_i \geq 0$ раз. Тогда длина последовательности есть $\sum_{i=1}^{m} k_i$. Предположим, что для любого целого $1 \leq i \leq m$ верно неравенство $k_i < n$. Тогда

$$\sum_{i=1}^{m} k_i \leq (n-1)m < nm.$$

Однако длина последовательности равна nm □

Лемма 3. Пусть m есть число вершин Dғрафа D. Ширина и глубина канона графа D ограничены числами $(w+1)m$ и $(d+1)m^2$ соответственно.

Доказательство. Предположим, что T является каноном, но

$$width(T) = s > (w+1)m.$$

Из определения ширины следует, что T содержит участок $T_1 \dots T_s$, в котором маршрут T_i, $i = 1, \dots, s$, непуст и нейтрален. Вследствие утверждения 2 неравенство $s > (w+1)m$ означает, что среди начальных вершин этих маршрутов по крайней мере $w+2$ одинаковы. Пусть, например, $beg(T_{i_j}) = beg(T_{i_{j+1}})$, где $1 \le i_j \le s$ и $1 \le j \le w+2$. Тогда $w+1$ участков $T_{i_j} \dots T_{i_{j+1}-1}$ являются нейтральными циклами. Следовательно, T не удовлетворяет первому условию определения канона вопреки допущению, что T есть канон.

Предположим теперь, что T — канон, но

$$depth(T) = k > (d+1)m^2.$$

Из определения глубины следует, что T содержит участок

$$T_{11} \dots T_{1k} T_{2k} \dots T_{21},$$

в котором T_{1i} и T_{2i}, $i = 1, \dots, k$, парны друг другу. Из неравенства $k > (d+1)m^2$ имеем существование таких номеров i_j, что $beg(T_{1i_j}) = beg(T_{1i_{j+1}})$ и $end(T_{2i_j}) = end(T_{2i_{j+1}})$ для $1 \le j \le d+2$. Тогда

$$T_{1i_j} \dots T_{1(i_{j+1}-1)} \text{ и } T_{2(i_{j+1}-1)} \dots T_{2i_j}$$

являются парными в T циклами для $1 \le j \le d+1$. Следовательно, T не удовлетворяет второму условию определения канона, вопреки допущению, что T есть канон □

Из теоремы 1 и леммы 3 вытекает

Теорема 3. Пусть m есть число вершин Dғрафа D. Длина канона графа D ограничена числом $g_{(w+1)m,(d+1)m^2}$. Следовательно, множество $Core(D)$ конечно □

Пример 8. Пусть M — магазинный автомат из примера 6, $D_2 = Graph(M)$. Тогда пересечение множеств $Core(D_2, 1, 1)$ и $Sentences(D_2)$ совпадает с

$$\{12^k 234^k 678(98)^l | k, l = 0, 1\} \cup \{1578(98)^l | l = 0, 1\}.$$

1.3.2. Теорема о развитии. Покажем, что ядро Dғрафа D выражает закон образования цепочек языка $L(D)$ из некоторых подцепочек пометок маршрутов, входящих в ядро. Для этого определим форманты — важный "материал строительства"произвольных маршрутов Dғрафа из присутствующих в ядре. Определяемое здесь же бинарное отношение развития на множестве маршрутов можно расценить как "инструмент строительства". Основная здесь теорема 4 имеет аналогию с классической теоремой о разрастании цепочек бесконтекстных языков.

Пусть (T_1, T_2, T_3) есть гнездо, в котором T_1 и T_3 — циклы. Пусть (T_{12}, T_2, T_{32}) и (T_{11}, $T_{12}T_2T_{32}$, T_{31}) не являются гнездами в маршруте $T_1T_2T_3$ для любых циклов T_{11}, T_{12}, T_{32}, T_{31}, таких, что $T_1 = T_{11}T_{12}$, $T_3 = T_{32}T_{31}$. Тогда назовем гнездо (T_1, T_2, T_3) *простым.*

Пусть маршрут T является нейтральным циклом или простым гнездом и не содержит меньшего участка, который есть нейтральный цикл или простое гнездо. Тогда назовем маршрут T *формантом.*

Определим бинарные отношения $\updownarrow_D$, $\uparrow_D$ на множестве маршрутов Dғрафа D: $(T', T) \in \updownarrow_D$ (с *историей* $\langle T_1, T_2, T_3, T_4, T_5\rangle$) тогда и только тогда, когда $T' = T_1T_3T_5$, $T = T_1T_2T_3T_4T_5$, $T_2T_3T_4$ есть формант, в котором T_2 и T_4 являются парными циклами; $(T', T) \in \uparrow_D$ (с *историей* $\langle T_1, T_2, end(T_2), end(T_2), T_5\rangle$) тогда и только тогда, когда $T = T_1T_2T_5$, $T' = T_1T_5$, T_2 есть циклический формант.

Объединение $\Uparrow_D = \updownarrow_D \cup \uparrow_D$ будем называть *отношением развития*.

Пусть T, T' — это маршруты графа D. Пусть либо $T' = T$, либо для некоторого $k > 0$ существует последовательность

$$gen(T) = (\langle T_{11}, T_{12}, T_{13}, T_{14}, T_{15}\rangle, \ldots, \langle T_{k1}, T_{k2}, T_{k3}, T_{k4}, T_{k5}\rangle),$$

такая, что $T' = T_{11}T_{13}T_{15}$, $T = T_{k1}T_{k2}T_{k3}T_{k4}T_{k5}$, $(T', T_{11}T_{12}T_{13}T_{14}T_{15}) \in \Uparrow_D$ с историей $\langle T_{11}, T_{12}, T_{13}, T_{14}, T_{15}\rangle$ и

$$(T_{(i-1)1}T_{(i-1)2}T_{(i-1)3}T_{(i-1)4}T_{(i-1)5}, T_{i1}T_{i2}T_{i3}T_{i4}T_{i5}) \in \Uparrow_D$$

с историей

$$\langle T_{i1}, T_{i2}, T_{i3}, T_{i4}, T_{i5}\rangle,\ 1 < i \le k.$$

Тогда назовем последовательность $gen(T)$ *родословной* (с *предком* T') маршрута T; число k назовем *длиной родословной*. Если $T = T'$, то родословная маршрута T (с предком T') пуста.

Отметим, что маршрут может иметь несколько родословных даже при одном и том же предке и одной и той же длине родословных.

Пусть маршрут T содержит нейтральный цикл или парные циклы. Тогда назовем маршрут T *производным*. Маршрут, не являющийся производным, назовем *элементарным*.

Теперь установим, что среди предков каждого нейтрального маршрута есть канон (более того, элементарный канон).

Лемма 4. Каждый производный маршрут содержит в себе формант.

Доказательство. Ясно, что каждый производный маршрут содержит производный нейтральный участок. Выберем среди нейтральных производных участков производного маршрута минимальный по длине. Предположим, что выбранный участок T не является формантом. Тогда из определения форманта следует, что T содержит меньший участок, который является нейтральным циклом или гнездом, образованным парными циклами. Итак, исходный маршрут содержит производный участок, который короче, чем T, вопреки выбору T □

Определим две функции, отображающие маршруты в множества маршрутов.

Пусть T — маршрут. Пусть

$$\mathcal{S}_1 = \{T_1T_3 \mid \exists \text{ (нейтральный цикл } T_2)\ T = T_1T_2T_3\},$$

$$\mathcal{S}_2 = \{T_1T_3T_5 \mid \exists \text{ (циклы } T_2, T_4)$$
$$(T_2, T_3, T_4) \text{ является гнездом в } T = T_1T_2T_3T_4T_5\}.$$

Тогда,

$$DelCycle(T) = \begin{cases} \{T\},\ \mathcal{S}_1 = \emptyset, \\ \mathcal{S}_1 \text{ иначе.} \end{cases}$$

$$DelPair(T) = \begin{cases} \{T\},\ \mathcal{S}_2 = \emptyset, \\ \mathcal{S}_2 \text{ иначе.} \end{cases}$$

Следующие функции отображают некоторые разбиения маршрутов во множества маршрутов.

Пусть $TT'T''$ — маршрут. Пусть $\mathcal{S}_3 = \{T_1T_3T_5 \in DelPair(TT'T'') \mid \exists\ (T_2,\ T_4,\ T_1',\ T_5')\ (T = T_1T_2T_1',\ T'' = T_5'T_4T_5,\ T_3 = T_1'T'T_5')\ \}$.

$$PreservingDel(T, T', T'') = \begin{cases} \{TT'T''\},\ \mathcal{S}_3 = \emptyset, \\ \mathcal{S}_3 \text{ иначе.} \end{cases}$$

Пусть $m \geq 1$, $T = T_1T_1' \ldots T_{m-1}T_{m-1}'T_m$ — маршрут. Пусть $\mathcal{S}$ обозначает объединение следующих двух множеств:

$$\{T_1T_1' \ldots T_{i-1}T_{i-1}'\hat{T}_iT_i' \ldots T_{m-1}T_{m-1}'T_m \mid 1 \leq i \leq m$$

и

$$\hat{T}_i \in DelCycle(T_i) \cup DelPair(T_i)\},$$

$$\{T_1T_1' \ldots T_{i-1}T_{i-1}'\hat{T}_iT_i' \ldots T_{j-1}T_{j-1}'\hat{T}_jT_j' \ldots T_{m-1}T_{m-1}'T_m | 1 \leq i < j \leq m,$$

$$\hat{T}_iT_i' \ldots T_{j-1}T_{j-1}'\hat{T}_j \in PreservingDel(T_i, T_i' \ldots T_{j-1}T_{j-1}', T_j)\}.$$

Если $\mathcal{S} = \emptyset$, то значением $reduction(T_1,\ T_1',\ \ldots,\ T_{m-1},\ T_{m-1}',\ T_m)$ является множество $\{T\}$, иначе множество

$$\cup_{\hat{T}_1T_1' \ldots \hat{T}_{m-1}T_{m-1}'\hat{T}_m \in \mathcal{S}} reduction(\hat{T}_1, T_1', \ldots, \hat{T}_{m-1}, T_{m-1}', \hat{T}_m).$$

Лемма 5. Пусть T — нейтральный маршрут; пусть $(T', T) \in \Uparrow_D$ с историей $\langle T_1,\ T_2,\ T_3,\ T_4,\ T_5\rangle$. Тогда существует (2,1)канон

$$T_0 = T_{01}T_2T_3T_4T_{05},$$

такой, что $pair(T_0) = pair(T)$.

Доказательство. Пусть

$$T_0 = T_{01}T_2T_3T_4T_{05} \in reduction(T_1, T_2T_3T_4, T_5).$$

Убедимся, что T_0 является (2,1)каноном. Из построения маршрута T_0 следует, что каждый его нейтральный цикл должен иметь общий участок с формантом $T_2T_3T_4$. Следовательно, для участка $T_{11} \ldots T_{1m}$ маршрута T_0, в котором T_{1i} является нейтральным циклом для $i = 1, \ldots, m$, число m не может превышать двух.

Из построения маршрута T_0 следует также, что хотя бы один из парных в нем циклов должен иметь общий участок с формантом $T_2T_3T_4$. Следовательно, гнездо $T' = T_2'T_3'T_4'$ маршрута T_0, которое отлично от $T_2T_3T_4$ и в котором T_2' и T_4' —парные циклы, содержит в себе формант $T_2T_3T_4$. Если $T_2T_3T_4$ содержится в одном из циклов T_2', T_4', то гнездо T' является простым. Если $T_2T_3T_4$ имеет общий участок с каждым из циклов T_2', T_4', то $pair(T_3) \neq pair(T_3')$ (иначе маршрут $T_2T_3T_4$ не был бы формантом). Следовательно, и в этом случае гнездо $T' \neq T_2T_3T_4$ является простым. Таким образом, T_0 удовлетворяет определению (2,1)канона $\square$

Прием применения функции $reduction$ подходит и в доказательстве следующего факта, который влечет возможность построения D-множества магазинного автомата за конечное число шагов.

Лемма 6. Пусть (π_1, T_2, π_2) есть гнездо некоторого предложения $T_1\pi_1T_2\pi_2T_3$, π_1 и π_2 являются дугами. Тогда существуют такие маршруты T_1', T_2', T_3', что $T_1'\pi_1T_2'\pi_2T_3'$ является предложением и (1,1)каноном, а (π_1, T_2', π_2) — его гнездом.

Доказательство. Рассмотрим маршрут

$$T' = T'_1\pi_1 T'_2\pi_2 T'_3 \in reduction(T_1, \pi_1, T_2, \pi_2, T_3).$$

Он, как и $T_1\pi_1 T_2\pi_2 T_3$, является предложением, так как удаление из маршрута нейтральных и парных циклов, осуществляемое операцией $reduction$, сохраняет его заряд и пару концов. По той же причине (π_1, T'_2, π_2) является гнездом маршрута T'. Предоставляем читателю проанализировать возможные положения в T' нейтральных и парных циклов и убедиться, что это (1,1)канон $\square$

Теорема 4 (о развитии маршрутов). Пусть D есть некоторый Dграф. Для каждого его нейтрального маршрута T существует такой канон T_0, что $(T_0, T) \in \Uparrow^*_D$. Для каждого элемента $\langle T_1, T_2, T_3, T_4, T_5\rangle$ родословной маршрута T с предком T_0 маршрут $T_2T_3T_4$ содержится в некотором каноне из $Core(D, 2, 1)$.

Доказательство. Достаточно рассмотреть производные маршруты. Пусть T — производный маршрут Dграфа D. По лемме 4 T содержит формант. Следовательно, возможны два случая:

(i) $T = T_1T_2T_3$ и T_2 есть формантцикл (тогда $(T_1T_3, T) \in \uparrow_D$);

(ii) $T = T_1T_2T_3T_4T_5$ и $T_2T_3T_4$ есть формант с парными в нем циклами T_2, T_4 (тогда $(T_1T_3T_5, T) \in \updownarrow_D$). В любом случае имеем такой маршрут T', что $(T', T) \in \Uparrow_D$. Так как $\mid T' \mid < \mid T \mid$, существует последовательность $T_0, \ldots, T_k$, такая, что $k > 0$, T_0 не содержит нейтральных и парных циклов, $T_k = T$, $(T_{i-1}, T_i) \in \Uparrow_D$ для $i = 1, \ldots, k$. Итак, $(T_0, T) \in \Uparrow^k_D$, T_0 является каноном, и первая часть теоремы верна. Вторая часть следует из леммы 5 $\square$

Ядро Dграфа, построенного по магазинному автомату M, будем называть также *ядром магазинного автомата* M и обозначать через $Core(M)$.

Пример 9. Пусть C есть подмножество (1,1)ядра Dграфа D_2, рассмотренное в примере 5. Его элемент

$$T = (1223467898)$$

содержит формантгнездо (2234) и формантцикл (98). Циклический участок (89), имеющий заряд $-a+a$, не является формантом. У маршрута T два непосредственных предка: $T_1 = (12367898)$ и $T_2 = (12234678)$. $T_3 = (123678)$ — общий предок маршрутов T_1 и T_2. Кроме маршрутов T, T_1, T_2, T_3, в C входит маршрут (157898) и его единственный предок (1578), совсем не содержащий циклов. Легко видеть, что (w, d)ядро рассматриваемого Dграфа, где w или d больше единицы, не определяет новых формантов.

Теорема о развитии означает, что ядро Dграфа D выражает закон образования цепочек языка $L(D)$ из некоторых подцепочек пометок маршрутов, входящих в ядро.

§2. О морфических представлениях бесконтекстного языка

Понятие Dграфа позволяет сравнительно легко получить теорему о морфическом представлении бесконтекстного языка, которая перекликается с теоремой ХомскогоШютценберже. Последняя выводится из нашей как простое следствие. Формулировка и доказательство нашей теоремы обобщают таковые для известной

теоремы о представлении регулярного языка морфическим образом локального языка [Salomaa 81].

Докажем нашу теорему конструктивно. При этом потребуется построение по Dграфу некоторых специфических локального языка и Dязыка. В целях этого построения введем несколько обозначений и докажем несложные леммы об условиях соединимости дуг и нейтральных маршрутов в большем маршруте. Входной Dграф обозначается через

$$D = (V, \Sigma, \mathcal{P}, \lambda, P_0, F).$$

Напомним, что множество нейтральных маршрутов Dграфа D является подмножеством Dязыка $\mathcal{L}_{\mathcal{P}}$.

Пусть Δ — алфавит, $x \in \Delta^+$, $x = ay = zb$ для некоторых $a, b \in \Delta$, $y, z \in \Delta^*$. Тогда

$$first(x) = a,$$
$$last(x) = b.$$

Пусть цепочка дуг $\pi\pi'$ является маршрутом. Тогда будем называть эти дуги *соединимыми* (*в порядке* $\pi\pi'$). Будем также говорить, что (π, π') есть *пара соединимых дуг*.

Важно отметить, что для построения всех пар соединимых дуг достаточно рассмотреть (2,2)ядро Dграфа D. Действительно, для каждого успешного маршрута $T_1\pi_1\pi_2T_2$, где $(\pi_1, \pi_2) \in \mathcal{P}$, любой элемент множества

$$reduction(T_1, \pi_1, end(\pi_1), \pi_2, T_2)$$

является, самое большее, (2,2)каноном.

Непосредственно из определений Dмножества $\mathcal{P}$, маршрута Dграфа и соединимых дуг получаем следующие две леммы.

Лемма 7. Пусть $(\pi_1, \pi_2) \in \mathcal{P}$ и T — нейтральный маршрут. Пусть

$$(\pi_1,\ first(T\pi_2)) \text{ и } (last(\pi_1 T),\ \pi_2)$$

суть пары соединимых дуг. Тогда $\pi_1 T \pi_2$ является нейтральным маршрутом □

Лемма 8. Пусть T_1 и T_2 — непустые нейтральные маршруты и

$$(last(T_1),\ first(T_2))$$

есть пара соединимых дуг. Тогда T_1T_2 является нейтральным маршрутом □

Пусть $E = E(D)$. Пусть

$$A = \{\pi \in E | beg(\pi) = P_0\},$$
$$B = \{\pi \in E | end(\pi) \in F)\},$$
$$C = \{\pi\pi' \in E^2 | \pi\pi' \text{ не является маршрутом}\}.$$

Тогда

$$R = (AE^* \cap E^*B) - E^*CE^*$$

является локальным языком.

Как показывает следующий пример, последовательность $\pi_1\pi_2$ дуг, удовлетворяющих равенству $end(\pi_1) = beg(\pi_2)$, не обязательно является маршрутом.

Пример 10. Граф совершенного магазинного автомата

$$(\{p_0, p_1, p_2, p_3\}, \{a, b\}, \{Z_0, a\}, Z_0, \delta, p_0, \{p_3\}),$$

где множество δ состоит из команд

$$(p_0, a, Z_0) \to (p_0, Z_0 a),$$
$$(p_0, a, a) \to (p_0, aa),$$
$$(p_0, b, a) \to (p_1, \Lambda),$$
$$(p_1, b, a) \to (p_1, \Lambda),$$
$$(p_1, \Lambda, Z_0) \to (p_2, Z_0 a),$$
$$(p_2, \Lambda, a) \to (p_3, \Lambda),$$

имеет Dмножество

$$\{(1,4),\ (1,6),\ (2,3),\ (2,5),\ (7,8)\}.$$

Здесь

$$\lambda(1) = (p_0, Z_0) a (p_0, a),$$
$$\lambda(2) = (p_0, a) a (p_0, a),$$
$$\lambda(3) = (p_0, a) b (p_1, a),$$
$$\lambda(4) = (p_0, a) b (p_1, Z_0),$$
$$\lambda(5) = (p_1, a) b (p_1, a),$$
$$\lambda(6) = (p_1, a) b (p_1, Z_0),$$
$$\lambda(7) = (p_1, Z_0) \Lambda (p_2, a),$$
$$\lambda(8) = (p_2, a) \Lambda (p_3, Z_0).$$

Последовательность $(2, 4)$ удовлетворяет условию $end(2) = beg(4)$, но не является маршрутом, так как вершина (p_1, Z_0) не может быть конечной вершиной маршрута с начальной памятью $(p_0, +a)$.

Лемма 9. Пусть $x = uvw \in \mathcal{L}_{\mathcal{P}} \cap R$, $\Lambda \neq v \in \mathcal{L}_P$. Тогда v является нейтральным маршрутом.

Доказательство. Индукцией по длине m непустой скобочной подсистемы v.

При $m = 2$ утверждение следует из определения локального множества R.

Пусть $k > 1$. Предположим, что утверждение верно для всех скобочных подсистем, длина которых не меньше двух и не больше $2(k-1)$, и рассмотрим в цепочке x скобочные подсистемы длины $2k$. Возможны следующие два случая.

Случай 1. Скобочная подсистема длины $2k$ имеет вид ayb, где $(a, b) \in \mathcal{P}$ и y является скобочной системой. По предположению индукции y является нейтральным маршрутом. Из соотношения $x \in \mathcal{L}_{\mathcal{P}} \cap R$ следует, что цепочка y и пара (a, b) удовлетворяют условиям леммы 7. Следовательно, ayb есть нейтральный маршрут.

Случай 2. Скобочная подсистема длины $2k$ имеет вид yz, где y и z сами являются непустыми скобочными системами. По предположению индукции y и z являются нейтральными маршрутами. Так как yz — подцепочка цепочки $x \in \mathcal{L}_{\mathcal{P}} \cap R$, пара $(last(y), first(x))$ есть пара соединимых дуг. Но тогда yz является нейтральным маршрутом по лемме 8 □

Теорема 5. Пусть $T \in \mathcal{L}_{\mathcal{P}}$. T является непустым успешным маршрутом тогда и только тогда, когда $T \in R$.

Доказательство. Легко видеть, что каждый непустой успешный маршрут удовлетворяет определению цепочки, входящей в R.

Пусть $T \in \mathcal{L}_{\mathcal{P}} \cap R$. Тогда по лемме 9 T является нейтральным маршрутом. Так как из соотношения $T \in R$ следуют соотношения $first(T) \in A$ и $last(T) \in B$, нейтральный маршрут T успешен $\square$

Теорема 6 (о морфическом представлении бесконтекстного языка). Пусть $L \subseteq \Sigma^*$ — бесконтекстный язык. Тогда существуют такие Dязык $\mathcal{L}$, локальный язык R и хороший (отображающий символ в символ или пустую цепочку [Eilenberg 74]) морфизм h, что

$$L = h(\mathcal{L} \cap R).$$

Доказательство. Так как L — бесконтекстный язык, он определяется некоторым Dграфом D. Используя множество дуг Dграфа D в качестве алфавита, определим локальный язык R, как в начале параграфа. Определим морфизм

$$h : E^* \to \Sigma^*$$

формулой

$$h(\pi) = \omega(\pi),\ \pi \in E.$$

По теореме 5 множество успешных маршрутов совпадает с $\mathcal{L}_{\mathcal{P}} \cap R$. Так как определяемый Dграфом язык есть множество пометок всех его успешных маршрутов, верна формула $L = h(\mathcal{L}_{\mathcal{P}} \cap R)$ $\square$

Следствие 1 (теорема ХомскогоШютценберже). Пусть $L \subseteq \Sigma^*$ — бесконтекстный язык. Тогда существуют такие язык Дика $\mathcal{L}$, локальный язык R и хороший морфизм h, что $L = h(\mathcal{L} \cap R)$.

Доказательство. Пусть $\mathcal{L}_P$ — это язык Дика над некоторым Dмножеством $\hat{p} \subset \Sigma_(\times \Sigma_)$, удовлетворяющим равенствам

$$|\Sigma_(| = |\Sigma_)| = |\hat{p}| = |\mathcal{P}|.$$

Пусть $\Delta = \Sigma_(\cup \Sigma_)$ и

$$\phi : \mathcal{P} \to \hat{p}$$

есть некоторая биекция. Можно сказать, что функция ϕ снабжает уникальным обозначением каждое вхождение в элементы Dмножества $\mathcal{P}$ скобок из $E = Left(\mathcal{P}) \cup Right(\mathcal{P})$.

Из определения функции ϕ следует сюръективность функции

$$\zeta : \Delta \to E,$$

заданной соотношениями (из равенств

$$|\Sigma_(| = |\Sigma_)| = |\hat{p}|$$

следует, что каждый символ алфавита Δ входит в некоторую пару из Dмножества $\hat{p}$):

$$\forall (a,b) \in \hat{p} \quad \zeta(a) = a',\ \zeta(b) = b',\ \phi(a',b') = (a,b).$$

Здесь возможно равенство $\zeta(c_1) = \zeta(c_2)$ при $c_1 \neq c_2$. Отметим, что функция ζ индуцирует побуквенный (отображающий символ одного алфавита в символ другого [Salomaa 81]) морфизм

$$\zeta : \Delta^* \to E^*.$$

Из определения функции ϕ следует биективность функции

$$\Phi : \mathcal{L}_{\mathcal{P}} \to \mathcal{L}_{\hat{p}}$$

определяемой соотношениями

$$\Phi(\Lambda) = \{\Lambda\},$$
$$\Phi(\pi_1 T \pi_2) = a\Phi(T)b,\ T \in \mathcal{L}_{\mathcal{P}},\ (\pi_1, \pi_2) \in \mathcal{P},\ (a, b) = \phi(\pi_1, \pi_2),$$
$$\Phi(T_1 T_2) = \Phi(T_1)\Phi(T_2),\ T_1, T_2 \in \mathcal{L}_{\mathcal{P}}.$$

Для произвольной подцепочки T цепочки из $\mathcal{L}_{\mathcal{P}}$ обозначим через $\Phi(T)$ множество $\{x_2|(\exists T_1, T_2, T_3)\ (T_2 = T,\ T_1 T_2 T_3 \in \mathcal{L}_{\mathcal{P}},\ \Phi(T_1 T_2 T_3) = x_1 x_2 x_3,\ |T_i| = |x_i|$ для $i = 1, 2, 3)\}$.

Пусть

$$A' = \bigcup_{\pi \in A} \Phi(\pi), \quad B' = \bigcup_{\pi \in B} \Phi(\pi), \quad C' = \Delta^2 - \bigcup_{\pi_1 \pi_2 \in E^2 - C} \Phi(\pi_1 \pi_2).$$

Тогда

$$R' = (A'\Delta^* \cap \Delta^* B') - \Delta^* C' \Delta^*$$

является локальным языком, а

$$\{\Phi(T)|T \text{ является успешным маршрутом D-графа } D\} = \mathcal{L}_P \cap R'.$$

Теперь видна справедливость равенства

$$L(D) = h(\zeta(\mathcal{L}_P \cap R')),$$

где h — морфизм, определенный в доказательстве теоремы 6. Так как композиция морфизмов есть морфизм и ζ — морфизм побуквенный,

$$h' = h\zeta : \Delta^* \to \Sigma^*$$

является хорошим морфизмом. Построенные язык Дика $\mathcal{L}_P$, локальное множество R' и хороший морфизм h' позволяют вывести из теоремы 6 справедливость рассматриваемого утверждения $\square$

§3. Одно обобщение регулярных выражений

В данном параграфе определяются и изучаются КС-выражения — обобщение регулярных выражений. Название отражает тот факт, что КС-выражения характеризуют бесконтекстные языки, или КС-языки. Вводится система понятий, которая служит инструментом исследования КС-выражений и отвечающих им языков. Эта система используется в данной статье для доказательства следующих двух фактов:

1) D-графы и КС-выражения характеризуют один и тот же класс языков;

2) класс так называемых псевдосогласующих КС-выражений эквивалентен классу регулярных выражений.

Отметим возможность несколькими способами доказывать, что КС-выражения характеризуют бесконтекстные языки. Первый из упомянутых выше фактов примечателен как обобщение теоремы Клини. Выделяется обширный подкласс КС-выражений, характеризующий регулярные языки.

Настоящий параграф следует считать продолжением параграфа 1, материал которого существенно здесь используется. В целом он представляет полученные к

данному моменту результаты наших исследований по выявлению и адекватному выражению общей сути различных характеризаций бесконтекстных языков. В пользу таких поисков можно заметить следующее. С помощью понятия синтаксической конгруэнции [Lallement 79] регулярный язык можно представить системой компонентов, не зависящей от тех или иных способов его задания, приписывающих его цепочкам некоторую структуру. При этом существует определенная взаимосвязь между компонентами упомянутой системы и любым из возможных синтаксисов регулярного языка (подразумеваются синтаксические описания, изучаемые в рамках теории регулярных языков). Именно наличие такой связи обеспечивает возможность отвечать на все вопросы о конечных автоматах.

Есть основания полагать, что некоторое обобщение известных результатов позволит раскладывать на компоненты детерминированные языки (а может быть, и бесконтекстные языки некоторых более общих классов) таким образом, чтобы это помогало отвечать на вопросы о соответствующих автоматах и грамматиках. В связи с этим ясно, что разложение устройств, задающих бесконтекстные языки, на некоторые компоненты помогает нащупать способ обобщения приемов, разработанных для более узких классов языков.

При разработке обобщения регулярных выражений мы руководствовались следующими соображениями. Звездочка Клини качественно отличается от других регулярных операций. Ее смысл позволяет интерпретировать регулярное выражение α, задающее бесконечный язык, как описание формальной системы, теоремы которой фиксируют "синтаксический разбор относительно α"отдельных цепочек языка. Для обоснования этого тезиса обсудим регулярные выражения над некоторым алфавитом Σ.

Язык, заданный регулярным выражением α, обозначим через $L(\alpha)$.

Заметим, что в записи регулярного выражения α над алфавитом Σ "рассредоточены"цепочки, составляющие некоторое конечное подмножество $\hat{L}(\alpha)$ языка $L(\alpha)$; $\hat{L}(\alpha)$ можно определить рекурсивно:

$\hat{L}(\emptyset) = \emptyset$,

$\hat{L}(a) = \{a\}$ для $a \in \Sigma \cup \{\epsilon\}$,

$\hat{L}(\alpha) = \hat{L}(\beta) \cup \hat{L}(\gamma)$ для $\alpha = \beta + \gamma$,

$\hat{L}(\alpha) = \hat{L}(\beta)\hat{L}(\gamma)$ для $\alpha = (\beta)(\gamma)$,

$\hat{L}(\alpha) = \{\Lambda\} \cup \hat{L}(\beta)$ для $\alpha = (\beta)^*$.

Теперь возьмем в качестве аксиомы само выражение α, а в качестве правила вывода удвоение вхождения подвыражения вида $(\gamma)^*$, т.е. посылке вида

$$x(\gamma)^* y,$$

где γ — регулярное выражение и xy — регулярное выражение или пустая цепочка, отвечает заключение

$$x(\gamma)^*(\gamma)^* y.$$

Ясно, что

$$L(\alpha) = \bigcup_{\beta \text{ выводится из } \alpha} \hat{L}(\beta).$$

Мы задались целью обобщить правило вывода рассмотренной выше системы, учитывая закон образования цепочек нерегулярного бесконтекстного языка, указанный вариантами теоремы о развитии [Станевичене 96б], [Станевичене 99] (в

настоящей работе им отвечают теоремы 2.4, 2.5 и 4). Кроме того, мы заботились о конечности числа подвыражений, которые разрешалось бы "тиражировать"при выводе из некоторых выражений других, более сложно устроенных. Потенциальная возможность добиться этого также указана теоремами о развитии. Таким образом, предлагаемое обобщение регулярных выражений обыгрывает формы разрастания цепочек, действующие в рамках классических характеризаций бесконтекстных языков.

В разделе 3.1 определяется КСвыражение. Определяется ряд понятий (ядро, развитие и т.д.), распространяющих наш подход к исследованию бесконтекстных языков на введенный новый формализм.

Раздел 3.2 рассматривает алгоритмы, преобразующие Dғраф в КСвыражение и КСвыражение в Dғраф. В разделе 3.3 определяются согласующее и псевдосогласующее КСвыражения. Доказывается утверждение об эквивалентности псевдосогласующих выражений регулярным.

3.1. КСвыражения. Весьма соблазнительна попытка построить обобщение регулярных выражений, которое задает в точности бесконтекстные языки и использует, кроме алфавита языка, некоторый не зависящий от языка конечный набор символов. Известная лемма Огдена, вид формантов, используемых для развития маршрутов Dғрафа, и вид элементарных итераций [Станевичене 96б] наталкивают на добавление к регулярным операциям операции, языкотворческий эффект которой таков же, как у системы бесконтекстных правил (вообще говоря, бесконечной)

$$S \to z|xSy,\ z \in Z \subseteq \Sigma^*,\ (x,y) \in X \times Y \subseteq \Sigma^* \times \Sigma^*.$$

Трудность, однако, в том, что языки Z, X и Y могут определяться с помощью языка L, порождаемого данными правилами, и места вставления парных цепочек $x \in X$ и $y \in Y$ не подчиняются какому-то общему для всех бесконтекстных языков закону. Места вставления различных "интерферирующих"конструкций могут располагаться в пределах некоторой цепочки так, что только индивидуальные их обозначения помогут разобраться, в каком месте что вставляется. В связи с этим вводится понятие именования. У операции именования два операнда: имя (это элемент некоторого конечного множества) и выражение, которому данное имя сопоставляется. Она старше используемых в КСвыражении регулярных операций объединения и сцепления.

Употребление именованного подвыражения внутри выражения с тем же именем обеспечивает размножение вхождений в цепочку языка некоторых подцепочек. Число используемых в выражении имен, вообще говоря, зависит от языка. Привычные средства описания синтаксической иерархии подцепочек в цепочках бесконтекстного языка побуждают выразить операцию именования посредством окаймления операндавыражения скобками $(_\iota$ и $)_\iota$, где ι — операндимя. С выбором нотации связана полезность при изучении КСвыражений рассмотренных в параграфе 1 комбинаторных свойств скобочных систем.

Так как скобочная система оказывается теперь "рассеянной"по цепочке, содержащей помимо скобок некоторые другие символы, часто будет употребляться следующее понятие проекции.

Для любых алфавитов Σ_1, Σ_2 и цепочки $x \in (\Sigma_1 \cup \Sigma_2)^*$ обозначим результат вычеркивания из x всех символов алфавита $\Sigma_1 - \Sigma_2$ через

$$projection(x, \Sigma_2).$$

Пусть $Names$ — непустое конечное множество, $\mathcal{B} = \{(_\iota,)_\iota | \iota \in Names\}$. Пусть множества Σ, $\mathcal{B}$ и $\{+, \epsilon, \emptyset, (,)\}$ попарно не пересекаются, $Alph = \Sigma \cup \mathcal{B} \cup \{+, \epsilon, \emptyset, (,)\}$. Определим рекурсивно *КСвыражение над алфавитом* Σ (это специфическое слово в алфавите $Alph$):

1) $a \in \Sigma \cup \{\epsilon, \emptyset\}$ — КСвыражение;

2) если α и β — КСвыражения, то:

a) $\alpha + \beta$ — КСвыражение; подвыражения α и β данного выражения будем называть его *слагаемыми*, само выражение — *суммой*;

b) $(\alpha)(\beta)$ — КСвыражение; подвыражения α и β выражения $(\alpha)(\beta)$ будем называть его *сомножителями*; заключать сомножитель в круглые скобки не обязательно, если он не является суммой;

3) если $\iota \in Names$ и β — КСвыражение, то $(_\iota \beta)_\iota$ есть (*именованное*) КСвыражение, или *ι-гнездо*.

Пусть множества $Names$, $\mathcal{B}$ и $Alph$ применяются в некотором КСвыражении ζ так, как это указано последним определением. Тогда обозначения $Alph(\zeta)$, $\mathcal{B}(\zeta)$ и $Names(\zeta)$ будут использоваться для множеств

$$\{c \in Alph | \exists (u, v \in Alph^*)\ \zeta = ucv\},$$

$$\mathcal{B} \cap Alph(\zeta)$$

и

$$\{\iota \in Names | (_\iota \in \mathcal{B}(\zeta)\}$$

соответственно.

Следующие определения фракции и клана помогают определить язык, задаваемый КСвыражением.

Назовем *фракцией* КСвыражения над алфавитом Σ элемент следующего рекурсивно определяемого множества КСвыражений:

1) $Fractions(a) = \{a\}$ для $a \in \Sigma \cup \{\epsilon, \emptyset\}$;

2) если α и β суть КСвыражения, то

$$Fractions((\beta)) = Fractions(\beta),$$

$$Fractions(\alpha + \beta) = Fractions(\alpha) \cup Fractions(\beta),$$

$$Fractions((\alpha)(\beta)) = Fractions(\alpha) Fractions(\beta),$$

$$Fractions((_\iota \beta)_\iota) = \{(_\iota\} Fractions(\beta) \{)_\iota\}.$$

Из определения фракции следует, что она выглядит как КСвыражение, которое не содержит сумм и безымянных скобок. На этом основании иногда уместно называть фракцией любую цепочку, которая имеет вид КСвыражения без сумм и безымянных скобок, не уточняя, какого КСвыражения эта фракция; в частности, мы будем называть фракцией каждый элемент определяемого далее клана КСвыражения.

Для фракции ζ над алфавитом Σ назовем ее *пометкой* следующее рекурсивно определяемое множество $\omega(\zeta) \subset \Sigma^*$:

$$\omega(\zeta) = \begin{cases} \{a\},\ \zeta = a \in \Sigma; \\ \{\Lambda\},\ \zeta = \epsilon; \\ \emptyset,\ \zeta = \emptyset; \\ \omega(\alpha)\omega(\beta),\ \zeta = \alpha\beta,\ \alpha \text{ и } \beta \text{ — фракции}; \\ \omega(\beta),\ \zeta = ({}_\iota\beta)_\iota,\ \beta \text{ — фракция}. \end{cases}$$

Заметим, что фракцию можно считать произведением, имеющим, возможно, только один множитель. Кроме того, заметим, что произведение множеств цепочек, хотя бы одно из которых пусто, считается пустым. Следовательно, каждая фракция, среди множителей которой есть $\emptyset$, имеет пустую пометку. Этим обстоятельством оправдывается привлечение следующего понятия приведенной фракции.

Пусть фракция совпадает с $\emptyset$ или не содержит подвыражения $\emptyset$. Тогда назовем фракцию *приведенной*.

Результат приведения некоторой фракции ξ обозначим через $trim(\xi)$. Приведение можно осуществить, повторяя, пока присутствуют гнездо вида $({}_\iota\emptyset)_\iota$ или произведение какого-либо из видов $\alpha\emptyset$, $\emptyset\alpha$, замену гнезда на ϵ, а произведения на $\emptyset$.

Обозначим через $Trim(\zeta)$ множество всех отличных от $\emptyset$ приведенных фракций КСвыражения ζ:

$$Trim(\zeta) = \{trim(\xi) | \xi \text{ — фракция выражения } \zeta\} - \{\emptyset\}.$$

Условимся, что если в некоторой формуле некоторые ее скобки выглядят парными друг другу, то они действительно парны в каждой цепочке, представляемой данной формулой. Это соглашение будет многократно использоваться. В частности, оно удобно в следующем определении.

Клан, порождаемый КСвыражением ζ, определим рекурсивно:

$$Clan(\zeta) = Trim(\zeta) \cup \{\beta_1({}_\iota\alpha_2)_\iota\beta_3 |$$

$$\alpha_1({}_\iota\alpha_2)_\iota\alpha_3, \beta_1({}_\iota\beta_2)_\iota\beta_3 \in Clan(\zeta)\}.$$

Из этого определения видно, что клан КСвыражения может отличаться от множества его приведенных фракций, если в КСвыражении несколько подвыражений одним и тем же именем. Действительно, для образования клана любое из именованных подвыражений может быть заменено любым одноименным. Замена, как увидим, "реализует"разрастание цепочек языка (см. ниже о понятиях цикла, развития и новом взгляде на образование клана).

Язык, задаваемый КСвыражением ζ, есть подмножество

$$L(\zeta) = \bigcup_{\xi \in Clan(\zeta)} \omega(\xi)$$

множества Σ^*. Язык пуст, если пуст клан.

Следующие определения легче понять, если вспомнить, что для любого КС-выражения ζ и для любого подмножества B алфавита именованных скобок $\mathcal{B}(\zeta)$

цепочка

$$projection(\zeta, B)$$

является скобочной системой.

Пусть $\iota \in Names(\zeta)$. Тогда назовем *ι-высотой* КСвыражения ζ число

$$h(\iota, \zeta) = depth(projection(\zeta, \{(_\iota,)_\iota\})).$$

Здесь $depth(x)$ обозначает глубину скобочной системы x, определенную в параграфе 1.

Назовем *высотой* КСвыражения ζ число

$$height(\zeta) = max\{h(\iota, \zeta) | \iota \in Names(\zeta)\}.$$

Подцепочки $(_\iota x$ и $z)_\iota$ КСвыражения

$$u(_\iota x(_\iota y)_\iota z)_\iota v$$

назовем его *парными друг другу циклами* (или, если информация об имени существенна, *ι-циклами*), *левым* и *правым* соответственно.

Заметим, что если $(_\iota x$ и $z)_\iota$ — парные циклы некоторой фракции и u — ιгнездо в ней, возможно, даже не содержащее этих циклов, то $(_\iota xuz)_\iota$ является ιгнездом в некотором элементе клана. Этим оправдывается целесообразность определяемого далее отношения развития.

Пусть $d \geq 0$. Назовем *d-ядром* КСвыражения ζ и обозначим через $Core(\zeta, d)$ подмножество всех элементов его клана, высота которых не превосходит d:

$$Core(\zeta, d) = \{\xi \in Clan(\zeta) | height(\xi) \leq d\}.$$

Покажем конечность множества $Core(\zeta, d)$.

Из определений фракции и клана следует, что для любой фракции $\xi \in Clan(\zeta)$ выполняется неравенство

$$width(projection(\xi, \mathcal{B}(\zeta))) \leq width(projection(\zeta, \mathcal{B}(\zeta))),$$

где $width(x)$ обозначает ширину (см. определение в параграфе 1) скобочной системы x. Кроме того, каждое подвыражение $\psi \in (\Sigma \cup \{\epsilon\})^*$ такой фракции не длиннее выражения ζ, т.е. между последовательными вхождениями скобок во фракцию расположена цепочка, длина которой ограничена длиной исходного КСвыражения. Из определения высоты КСвыражения следует, что для любой фракции $\xi \in Clan(\zeta)$ глубина

$$depth(projection(\xi, \mathcal{B}(\zeta)))$$

не превосходит произведения $d \cdot |Names(\zeta)|$. Значит, множество $Core(\zeta, d)$ конечно вследствие теоремы 1 (она же есть теорема 1 работы [Станевичене 99]).

Пусть ιгнездо u входит в некоторый элемент клана КСвыражения ζ и удовлетворяет условиям $h(\iota, u) = 2$, $h(o, u) < 2$ для любого имени $o \neq \iota$. Тогда назовем u *формантом выражения* ζ.

Лемма 10 (см. ниже) показывает, что все форманты КСвыражения обнаруживаются в элементах его 2ядра.

Определим на множестве $Clan(\zeta)$ отношение *непосредственного развития* $\Uparrow_\zeta$: пусть $u = v({}_\iota x({}_\iota y)_\iota z)_\iota w \in Clan(\zeta)$, где $({}_\iota x({}_\iota y)_\iota z)_\iota$ является формантом. Тогда

$$(v({}_\iota y)_\iota w, u) \in \Uparrow_\zeta .$$

Пятерку

$$(v, ({}_\iota x, ({}_\iota y)_\iota, z)_\iota, w)$$

будем называть *историей непосредственного развития выражения* u *из* $v({}_\iota y)_\iota w$.

Отношение $\Uparrow_\zeta^*$ назовем *отношением развития.*

Уподобляя циклы КСвыражения таковым Dғрафа, нетрудно определить аналог функции $reduction$, отображающий разложение КСвыражения в множество КСвыражений и вычеркивающий в исходном выражении циклы, которые можно вычеркнуть, не затрагивая некоторых выделенных участков выражения. Этой новой функции также дадим имя $reduction$.

Более аккуратно, пусть $k \geq 1$, а $\zeta = x_1 \dots x_k$ — КСвыражение, части x_j которого могут и не быть выражениями. Обозначим через $reduction(x_1, \dots, x_k)$ множество всех КСвыражений, получаемых из ζ такими удалениями (они делаются, пока возможны) парных циклов, что части x_{2i}, $2 \leq 2i \leq k$, не затрагиваются.

Лемма 10. Если $\alpha\beta\gamma \in Clan(\zeta)$, а β является формантом, то $reduction(\alpha, \beta, \gamma) \subseteq Core(\zeta, 2)$.

Доказательство. Пусть

$$\alpha'\beta\gamma' \in reduction(\alpha, \beta, \gamma),$$

где α' и γ' являются образами цепочек α и γ при данном редуцировании. Предположим, что для некоторого имени ι выражение $\alpha'\beta\gamma'$ содержит ιғнездо ξ ιвысоты 3. Заметим, что границами гнезда являются парные друг другу скобки. Следовательно, если гнезда u и v суть различные участки одного КСвыражения, то либо они не имеют общего участка, либо одно из гнезд является участком другого. Отсюда получаем, что гнездо ξ содержит в себе формант β, так как внутри форманта не может быть гнезда столь большой высоты, а наличие гнезда ξ вне β противоречит построению выражения $\alpha'\beta\gamma'$. Но тогда, опятьтаки вследствие определения форманта, α' и γ' должны соответственно содержать левый и правый циклы гнезда ξ, что противоречит построению выражения $\alpha'\beta\gamma'$ □

Лемма 11. Если фракция $\xi \in Clan(\zeta)$ содержит циклы, то существует такая фракция $\xi' \in Clan(\zeta)$, что $(\xi', \xi) \in \Uparrow_\zeta$ и $|\xi'| < |\xi|$.

Доказательство. Докажем, что ξ содержит формант. Среди ее гнезд, принадлежащих множеству $Nests = \{u | \exists(v, w \in Alph(\zeta)^*,\ \iota \in Names(\zeta))\ (\xi = vuw,\ u$ — ιғнездо, $h(\iota, u) \geq 2)\}$, выберем выражение с наименьшей длиной. В силу выбора оно не содержит собственных подвыражений, принадлежащих множеству $Nests$, т.е. является формантом. Пусть $\xi = \alpha\beta\gamma$, где $\beta = ({}_\iota x({}_\iota y)_\iota z)_\iota$ — выбранный формант. Тогда согласно определению отношения непосредственного развития для $\xi' = \alpha({}_\iota y)_\iota \gamma$ имеем $(\xi', \xi) \in \Uparrow_\zeta$ и $|\xi'| < |\xi|$ □

Теорема 7. Пусть ζ есть КСвыражение. Для любого выражения $\xi \in Clan(\zeta)$ существуют такие число $k \geq 0$ и последовательность КСвыражений $\xi_0, \dots, \xi_k$, что $\xi_0 \in Core(\zeta, 1)$, $\xi_k = \xi$ и $(\xi_{i-1}, \xi_i) \in \Uparrow_\zeta$ для $0 < i \leq k$ с некоторой историей

непосредственного развития $(u_i, x_i, y_i, z_i, v_i)$, причем формант $x_i y_i z_i$ содержится в некотором элементе ядра $Core(\zeta, 2)$.

Доказательство. Для $\xi \in Core(\zeta, 1)$ теорема верна при $k = 0$. Пусть теперь $\xi \in Clan(\zeta) - Core(\zeta, 1)$. Тогда ξ содержит циклы. По лемме 11 существует такое выражение $\xi' \in Clan(\zeta)$, что $(\xi', \xi) \in \Uparrow_\zeta$ и $|\xi'| < |\xi|$. Из последнего неравенства следует, что существуют $k \geq 1$ и последовательность $\xi_0, \ldots, \xi_k$, такие, что ξ_0 не содержит циклов (т.е. входит в $Core(\zeta, 1)$), $\xi_k = \xi$ и $(\xi_{i-1}, \xi_i) \in \Uparrow_\zeta$ для $0 < i \leq k$ с некоторой историей непосредственного развития $(u_i, x_i, y_i, z_i, v_i)$. По лемме 10 формант $x_i y_i z_i$ содержится в некотором элементе ядра $Core(\zeta, 2)$ $\square$

Пример 11. Язык

$$\{a\} \cup \{a^m b^n | m \geq 1, n \geq m\}$$

может быть задан КСвыражением

$$\zeta = a + ({}_1a({}_1ab)_1b)_1({}_2({}_2\epsilon)_2b)_2.$$

Наибольшие два из четырех его гнезд являются формантами.

$$Clan(\zeta) = \{a\} \cup \{[({}_1a]^k({}_1ab)_1[b)_1]^k[({}_2]^l({}_2\epsilon)_2[b)_2]^l | k, l \geq 0\}.$$

Простейшими элементами этого клана являются не содержащие циклов фракции a и $({}_1ab)_1({}_2\epsilon)_2$. Объединение их пометок $\{a, ab\}$ содержит две самые короткие цепочки языка $L(\zeta)$. Для любого целого $d > 1$

$$Core(\zeta, d) = \{a\} \cup \{[({}_1a]^k({}_1ab)_1[b)_1]^k[({}_2]^l({}_2\epsilon)_2[b)_2]^l | 0 \leq k, l \leq d\}.$$

3.2. Эквивалентность КСвыражений и Dғрафов. Следующий прием перевода регулярных выражений в КСвыражения специального вида показывает, что последние можно считать формой записи регулярных выражений и что приведенная далее теорема 8 — достаточно естественное обобщение теоремы Клини.

Пусть α — регулярное выражение над алфавитом Σ. Тогда $L(\alpha) = L(\sigma(\alpha))$, где КСвыражение $\sigma(\alpha)$ определяется рекурсивно следующим образом:

1) $\sigma(a) = a$, $a \in \Sigma \cup \{\epsilon, \emptyset\}$;

2) если β и γ — регулярные выражения, то $\sigma(\beta + \gamma) = \sigma(\beta) + \sigma(\gamma)$, $\sigma((\beta)(\gamma)) = (\sigma(\beta))(\sigma(\gamma))$ и $\sigma((\beta)^*) = ({}_\beta\sigma(\beta)({}_\beta\epsilon)_\beta)_\beta$.

Теорема 8. Язык задается КСвыражением тогда и только тогда, когда он определяется некоторым Dғрафом.

Дадим конструктивное доказательство теоремы. Для этого потребуется несколько новых понятий и обозначений.

Для Dғрафа используем наше обычное обозначение

$$D = (\Sigma,\ V,\ \mathcal{P},\ \lambda,\ P_0,\ F).$$

Назовем *праволинейным разложением* непустого нейтрального маршрута T четверку

$$rlf(T) = (\pi_1, T_1, \pi_2, T_2),$$

где $(\pi_1, \pi_2) \in \mathcal{P}$, $parti(\pi_1 T_1 \pi_2) = 1$, $T = \pi_1 T_1 \pi_2 T_2$.

Праволинейное разложение маршрута определяется однозначно, так как у маршрута не может быть двух начальных участков протяжения 1.

Если $rlf(T) = (\pi_1, T_1, \pi_2, T_2)$, то $(1,1)$ядро Dғрафа содержит маршрут с праволинейным разложением $(\pi_1, T_1', \pi_2, T_2')$, где $pair(T_i') = pair(T_i)$ для $i = 1, 2$. (Действительно (ср. лемму 6), таков каждый маршрут из $reduction(\pi_1, T_1, \pi_2, T_2) \subseteq Core(D, 1, 1)$.) Как будет ясно из дальнейшего, это означает, что $(1,1)$ядро D-графа содержит всю необходимую информацию для осуществления предлагаемых ниже построений. Далее пишем $Core(D)$ вместо $Core(D, 1, 1)$.

Назовем *фразой* Dғрафа D элемент следующего рекурсивно определяемого множества нейтральных маршрутов: $Phrases(D) = Sentences(D) \cup \{T_1, T_2 | \exists((\pi_1, \pi_2) \in \mathcal{P},\ T \in Phrases(D))\ (\pi_1, T_1, \pi_2, T_2) = rlf(T)\}$.

Построим по Dғрафу D КСвыражение $\mathcal{E}(D)$ над алфавитом дуг $E(D)$. Для индексации скобок, составляющих алфавит $\mathcal{B}(\mathcal{E}(D))$, используем элементы множества

$$\{pair(T) | T \in Phrases(D)\}.$$

В следующих множествах фраз каждый элемент, который является пустым маршрутом, следует считать синонимом обозначения Λ, а сами множества — языками в алфавите $E(D)$:

$$Alt(P, Q) = \{T \in Phrases(D) | pair(T) = (P, Q)\},$$

$$Productions(P, Q) = Alt(P, Q) \cap Core(D).$$

Пусть $T \in Phrases(D)$. Тогда *схема* $scheme(T)$ фразы T определяется рекурсивно следующим образом:

1) если фраза T пуста, то $scheme(T) = ({}_{pair(T)}\epsilon)_{pair(T)}$;

2) если $rlf(T) = (\pi_1, T_1, \pi_2, T_2)$, то

$$scheme(T) = ({}_{pair(T)}\pi_1 scheme(T_1)\pi_2 scheme(T_2))_{pair(T)}.$$

Ясно, что схема удовлетворяет определению фракции над алфавитом $E(D)$. Будем считать, что пометкой фракции $({}_{pair(T)}\epsilon)_{pair(T)}$ является $\{T\}$. Хотя пустых фраз может быть много, нужный синоним обозначения Λ однозначно устанавливается по индексу $pair(T)$ скобок данной фракции. Действительно, вершина $beg(T) = end(T)$ и является представлением пустого маршрута T. Это соглашение о пометке удобно в доказательстве нижеследующей леммы 3.

Конструкция выражения $\mathcal{E}(D)$ очень проста:

$$\mathcal{E}(D) = \sum_{Q \in F} \sum_{T \in Productions(P_0, Q)} scheme(T).$$

Из определения фразы следует, что $Alt(P, Q)$ есть множество $\{\pi_1 T_1 \pi_2 T_2 | beg(\pi_1) = P;\ parti(\pi_1 T_1 \pi_2) = 1;\ T_1 \in Alt(end(\pi_1), beg(\pi_2));\ T_2 \in Alt(end(\pi_2), Q)\}$, объединенное с $\{P\}$ при $P = Q$.

Пусть

$$\tau(P, Q) = \{scheme(T) | T \in Productions(P, Q)\}.$$

Обозначим через $L_{P,Q}$ язык, задаваемый суммой $\sum_{\xi \in \tau(P,Q)} \xi$. Ясно, что

$$L_{P,Q} \subseteq Alt(P, Q).$$

Докажем, что

$$Alt(P, Q) \subseteq L_{P,Q}.$$

Для этого докажем следующую лемму.

Лемма 12. Пусть T — фраза. Тогда $T \in L_{pair(T)}$.

Доказательство. Пусть

$$\zeta = \sum_{\xi \in \tau(pair(T))} \xi.$$

Заметим, что из построения схем следуют равенства

$$Fractions(\zeta) = Trim(\zeta) = \tau(pair(T)).$$

Вследствие теоремы 7 достаточно доказать, что $T \in \omega(\psi)$, где $(\xi, \psi) \in \Uparrow_\zeta^m$ для некоторых $\xi \in \tau(pair(T))$ и $m \geq 0$. Применим индукцию по $k \geq 0$, где $2k$ — длина фразы T.

При $k = 0$ фраза T пуста. Следовательно,

$$\xi = ({}_{pair(T)}\epsilon)_{pair(T)} \in \tau(pair(T)).$$

Так как $(\xi, \xi) \in \Uparrow_\zeta^0$ и $T \in \omega(\xi)$, утверждение в данном случае верно.

Пусть $k > 0$. Предположим, что утверждение справедливо для всех фраз, длина которых меньше, чем $2k$, и рассмотрим фразу T длины $2k$. Допустим, что

$$rlf(T) = (\pi_1, T_1, \pi_2, T_2).$$

Из предположения индукции следует, что для $i = 1, 2$ верно соотношение $T_i \in \omega(\psi_i)$, где $(\xi_i, \psi_i) \in \Uparrow_\zeta^{m_i}$ для некоторых $\xi_i \in \tau(pair(T_i))$ и $m_i \geq 0$. Можно считать (ср. соотношение для ξ_0 в теореме 7), что $height(\xi_i) = 1$, $i = 1, 2$. Но тогда

$$\xi = \pi_1 \xi_1 \pi_2 \xi_2 \in \tau(pair(T)).$$

Так как

$$(\xi, \pi_1 \psi_1 \pi_2 \psi_2) \in \Uparrow_\zeta^{m_1 + m_2}$$

и

$$T = \pi_1 T_1 \pi_2 T_2 \in \omega(\pi_1 \psi_1 \pi_2 \psi_2),$$

утверждение верно и при $k \geq 0$.

Итак, $T \in L_{pair(T)}$ $\square$

Из леммы следует равенство $L(\mathcal{E}(D)) = Sentences(D)$.

Заметим, что $L(D)$ есть совокупность пометок всех маршрутов, которые входят в $Sentences(D)$.

Определим морфизм φ соотношениями:

$$\varphi(\pi) = \omega(\pi),\ \pi \in E(D);\ \varphi(a) = a,\ a \notin E(D).$$

Тогда ясно, что КСвыражение $Expr(D) = \varphi(\mathcal{E}(D))$ задает язык $L(D)$. Это означает, что любой бесконтекстный язык определяется некоторым КСвыражением.

Пример 12. Пусть D есть граф магазинного автомата M, рассмотренного в примере 6. Вместо выражения $Expr(D)$, страдающего избыточностью, приведем несколько менее избыточное эквивалентное выражение ζ, построенное по следующим двум элементам множества $Productions((p_0, Z_0), (f, Z_0))$: 157898, 12234678.

Пары концов фраз, входящих в эти успешные маршруты, дают семь имен: 1) $((p_0,\ Z_0),\ (f,\ Z_0))$, 2) $((p_0, a),\ (p_0, a))$, 3) $((p_1, Z_0),\ (f, Z_0))$, 4) $((p_2, a),\ (p_2, a))$, 5) $((f, Z_0),\ (f, Z_0))$, 6) $((p_0, a),\ (p_1, a))$, 7) $((p_1, a),\ (p_1, a))$. В выражении ζ будем писать вместо перечисленных имен их порядковые номера:

$$\zeta = ({}_1a({}_2\epsilon)_2b({}_3({}_4\epsilon)_4({}_5b({}_4\epsilon)_4({}_5\epsilon)_5)_5)_3)_1+$$

$$({}_1a({}_6a({}_6a({}_2\epsilon)_2b({}_7\epsilon)_7)_6b({}_7\epsilon)_7)_6b({}_3({}_4\epsilon)_4({}_5\epsilon)_5)_3)_1.$$

Символы a, b, перемежающие собой именованные скобки, отвечают пометкам дуг; ϵ отвечает пустому участку.

Теперь докажем, что каждое КСвыражение определяет некоторый бесконтекстный язык. Достаточно рассмотреть случай, когда клан КСвыражения непуст; в противном случае выражение задает пустой язык, относящийся к бесконтекстным.

Сначала преобразуем КСвыражение (пусть это ζ) к виду, удобному для построения эквивалентного выражению магазинного автомата. В нашем построении важна роль гнезд; им соответствуют четко обособленные подавтоматы. Добьемся некоторой стандартной организации гнезд, входящих в элементы клана, вводя, если нужно, дополнительные имена. Можно заранее рассчитывать, что любая фракция в клане является произведением не менее чем двух сомножителей, а для каждого гнезда $({}_\iota\beta)_\iota$, которое входит в некоторую фракцию из $Clan(\zeta)$, либо $\beta \in \Sigma \cup \{\epsilon\}$, либо β является произведением по крайней мере двух сомножителей. Действительно, для выполнения этих требований достаточно добавить сомножители ϵ к каждой фракции из $Fractions(\zeta)$, а каждое входящее в такую фракцию гнездо вида

$$({}_{\iota_1}({}_{\iota_2}\beta)_{\iota_2})_{\iota_1},$$

где $\beta \in Alph(\zeta)^+$, $\iota_1, \iota_2 \in Names(\zeta)$, заменить на

$$({}_{\iota_1}\epsilon({}_{\iota_2}\beta)_{\iota_2})_{\iota_1}.$$

Пусть КСвыражение либо является именованным, либо совпадает с некоторым $a \in \Sigma \cup \{\epsilon, \emptyset\}$. Тогда назовем это выражение *неделимым*.

Определяемая непосредственно далее разметка фактически ранжирует неделимые сомножители внутри отдельных гнезд и во фракциях в целом. Вводимая разметкой упорядоченность сомножителей позволяет отвлечься от ассоциативности умножения и вследствие этого уменьшить размеры автомата, производимого по КСвыражению. Но главное назначение разметки — решить проблему различения вхождений одних и тех же участков в рассматриваемые фракции. Так, для любой фракции $xay \in Clan(Augm(\zeta))$, где $a \in \Sigma \cup \{\epsilon\}$, а $Augm(\zeta)$ определяется (см. ниже) по КСвыражению ζ над алфавитом Σ с помощью разметки, цепочка $projection(x, \mathcal{B}(Augm(\zeta)))$ уникальна, т.е. может использоваться для идентификации рассматриваемого вхождения a. Эта особенность весьма облегчает доказательство корректности нашего построения магазинного автомата по КСвыражению.

Пусть $\xi \neq \emptyset$ приведенная фракция, κ — конечная последовательность некоторых натуральных чисел. Назовем *разметкой фракции* ξ (*с помощью* κ) и обозначим через $rlm(\xi, \kappa)$ следующую рекурсивно определяемую фракцию:

1) если $a \in \Sigma \cup \{\epsilon\}$, то

$$rlm(a, \kappa) = ({}_\kappa a)_\kappa$$

и

$$rlm(({}_\iota a)_\iota, \kappa) = ({}_\iota a)_\iota$$

для любого имени ι;

2) если α является неделимым КСвыражением, то

$$rlm(\alpha\beta, \kappa) = ({}_\kappa rlm(\alpha, (\kappa, 1))rlm(\beta, (\kappa, 2)))_\kappa,$$

$$rlm(({}_\iota\alpha\beta)_\iota, \kappa) = ({}_\iota rlm(\alpha, (\kappa, 1))rlm(\beta, (\kappa, 2)))_\iota$$

для любых фракции β и имени ι.

Как видим, любые две пары скобок, добавленные разметкой, индексируются различными именами.

Пусть клан КСвыражения ζ непуст; пусть цифра 0 и кортежи натуральных чисел не используются в ζ как имена гнезд. Занумеруем элементы множества $Core(\zeta, 2)$ в некотором порядке и обозначим через $\nu(\xi)$ номер фракции $\xi \in Core(\zeta, 2)$. С помощью размеченных фракций 2ядра определим множество $Mark(\zeta)$ следующим образом:

$$Mark(\zeta) = \{rlm(({}_0\xi)_0, \nu(\xi))|\xi \in Core(\zeta, 2)\}.$$

Назовем *пополнением* КСвыражения ζ сумму фракций, составляющих множество $Mark(\zeta)$:

$$Augm(\zeta) = \sum_{\xi \in Mark(\zeta)} \xi.$$

Следующая лемма используется в доказательстве эквивалентности выражений ζ и $Augm(\zeta)$.

Лемма 13. Пусть клан КСвыражения ζ непуст. Тогда:

1) $Core(\zeta, 1) = \{projection(\xi, Alph(\zeta))|\xi \in Core(Augm(\zeta), 1)\}$;

2) множество формантов КСвыражения ζ совпадает с

$$\{projection(\xi, Alph(\zeta))|\xi \text{ — формант выражения } Augm(\zeta)\}.$$

Доказательство. Заметим, что $h(\kappa, \psi) \leq 1$ для $\psi \in Mark(\zeta)$ и любого имени κ, введенного разметкой, которая превращает множество $Core(\zeta, 2)$ в $Mark(\zeta)$, т.е. для $\kappa \in Names(Augm(\zeta)) - Names(\zeta)$. Следовательно, каждый формант, участвующий в некоторой фракции из $Core(\zeta, 2)$, превращается в формант ее разметки. Других формантов фракции из $Mark(\zeta)$ не содержат.

Отсюда ясно, что $Core(Augm(\zeta), 1)$ совпадает с

$$\{\xi \in Mark(\zeta)|projection(\xi, Alph(\zeta)) \in Core(\zeta, 1)\}.$$

Так как

$$Core(\zeta, 1) \subseteq Core(\zeta, 2) = \{projection(\xi, Alph(\zeta))|\xi \in Mark(\zeta)\},$$

первое из утверждений леммы справедливо.

Из последнего равенства вытекает совпадение множества формантов выражения ζ с множеством

$$FS = \{projection(\xi, Alph(\zeta))|\xi \text{ — формант},$$

$$\exists(u, v \in Alph(Augm(\zeta))^*)\ u\xi v \in Mark(\zeta)\}.$$

Вообще говоря, фракции из $Mark(\zeta)$ содержат не все форманты выражения $Augm(\zeta)$. В самом деле, если в них есть одноименные форманты

$$({}_\iota x_1({}_\iota y_1)_\iota z_1)_\iota \text{ и } ({}_\iota x_2({}_\iota y_2)_\iota z_2)_\iota,$$

где $x_1y_1z_1 \neq x_2y_2z_2$, то в элементах клана $Clan(Augm(\zeta))$ присутствуют форманты

$$({}_\iota x_1({}_\iota y_2)_\iota z_1)_\iota \text{ и } ({}_\iota x_2({}_\iota y_1)_\iota z_2)_\iota.$$

Однако по лемме 11 проекции всех четырех рассмотренных формантов на $Alph(\zeta)$ присутствуют в элементах 2ядра выражения ζ. Таким образом, множество FS содержит проекции на $Alph(\zeta)$ всех формантов выражения $Augm(\zeta)$, одноименных с формантами выражения ζ. Следовательно, для доказательства второго из утверждений леммы теперь достаточно убедиться, что выражение $Augm(\zeta)$ не имеет формантов с именами из $Names(Augm(\zeta)) - Names(\zeta)$.

Если $\kappa \in Names(Augm(\zeta)) - Names(\zeta)$, то любое κғнездо κғлубины 2 не является формантом. Действительно, вследствие уникальности каждой пары скобок, поставленной при разметке, гнездо v вида $({}_\kappa x({}_\kappa y)_\kappa z)_\kappa$ может появиться в элементе множества $Clan(Augm(\zeta)) - Mark(\zeta)$ только в результате последовательного использования для развития фракций из $Mark(\zeta)$ некоторых формантов

$$({}_\iota u_1({}_\kappa u_2({}_\iota u_3)_\iota u_4)_\kappa u_5)_\iota,\ ({}_\iota u_1({}_\kappa u_2({}_\iota u'_3)_\iota u_4)_\kappa u_5)_\iota$$

с одними и теми же парными циклами. Следовательно, ιғлубина гнезда v не меньше двух, и оно не удовлетворяет определению форманта.

Таким образом, элементы из $Names(Augm(\zeta)) - Names(\zeta)$ играют роль индивидуальных имен подвыражений фракций из $Core(\zeta, 2)$; на образование клана выражения $Augm(\zeta)$ влияют только гнезда с именами из $Names(\zeta)$, отличающиеся от гнезд множества $Core(\zeta, 2)$ лишь присутствием "индивидуализирующих скобок". Следовательно, верно и второе из утверждений леммы □

Лемма 14. Выражения ζ и $Augm(\zeta)$ эквивалентны.

Доказательство. Достаточно доказать, что клан $Clan(Augm(\zeta))$ сюръективно отображается на $Clan(\zeta)$ и образом фракции $\xi \in Clan(Augm(\zeta))$ является

$$projection(\xi, Alph(\zeta)) \in Clan(\zeta).$$

Но этот факт вытекает непосредственно из нижеследующего утверждения, которое легко доказать индукцией по k, опираясь на установленные в лемме 13 взаимосвязи ядер и формантов выражений ζ и $Augm(\zeta)$.

Пусть $k \geq 0$. Тогда

$$\{projection(\xi, Alph(\zeta)) | \exists \xi_0 \in Core(Augm(\zeta), 1)\ (\xi_0, \xi) \in \Uparrow^k_{Augm(\zeta)}\} =$$

$$\{\psi | \exists \psi_0 \in Core(\zeta, 1)\ (\psi_0, \psi) \in \Uparrow^k_\zeta\}$$

□

Теперь введем термины, касающиеся особенностей выражения $Augm(\zeta)$ и подготавливающие построение по нему эквивалентного магазинного автомата.

Будем называть *активным гнездом* любое именованное выражение, входящее в некоторую фракцию из $Clan(Augm(\zeta))$.

Пусть $k, l, m \in Names(Augm(\zeta))$, $\alpha, \beta \in Alph(Augm(\zeta))^*$ и $\xi = ({}_m({}_k\alpha)_k({}_l\beta)_l)_m$ — активное гнездо. Тогда назовем тройку (k, l, m) *альянсом* и будем говорить, что ξ *представляет этот альянс.*

Заметим, что по построению разметки каждое активное гнездо либо представляет некоторый альянс, либо имеет вид

$$({}_\iota a)_\iota,\ a \in \Sigma \cup \{\epsilon\},\ \iota \in Names(Augm(\zeta)).$$

Лемма 15. Пусть $({}_m({}_k\alpha)_k({}_l\beta)_l)_m$ — активное гнездо. Тогда существует активное гнездо $({}_m({}_k\alpha')_k({}_l\beta')_l)_m$, которое входит в некоторую фракцию из $Core(Augm(\zeta),3)$.

Доказательство. Согласно определению активного гнезда существуют такие u, $v \in Alph(Augm(\zeta))^*$, что

$$u({}_m({}_k\alpha)_k({}_l\beta)_l)_m v \in Clan(Augm(\zeta)).$$

kВысота фракции

$$\xi' \in reduction(u, ({}_m({}_k, \alpha)_k, ({}_l, \beta)_l)_m v)$$

может достигать трех. Действительно, пусть ξ имеет вид

$$u_1({}_k u_2({}_m({}_k\alpha_1({}_k\alpha_2)_k\alpha_3)_k({}_l\beta)_l)_m v_1)_k v_2.$$

Тогда никакая из показанных пар kциклов не может исчезнуть при редуцировании, так как каждый из данных kциклов содержит некоторую сохраняемую часть редуцируемого выражения.

Нетрудно убедиться, что для любых других допустимых случаев положения скобок $({}_k$ и $)_k$ неравенство $h(k,\xi') > 3$ не выполняется. Аналогично, $h(l,\xi') \le 3$. Для любого другого имени $\iota \in Names(Augm(\zeta))$ $h(\iota,\xi') \le 2$ $\square$

Лемма 15 означает, что любой альянс представлен некоторым гнездом, входящим в выражение из $Core(Augm(\zeta),3)$. Это оправдывает следующее использование множества $Core(Augm(\zeta),3)$ для построения эквивалентного КСвыражению ζ магазинного автомата

$$M(\zeta) = (\mathcal{B}(Augm(\zeta)),\ \Sigma, Names(Augm(\zeta)) \cup \{Z_0\}, Z_0, \delta(\zeta), ({}_0, \{)_0\}).$$

Так как доказательство эквивалентности выражений $Augm(\zeta)$ и $M(\zeta)$ удобно вести в терминах маршрутов, построим вместо $\delta(\zeta)$ Dмножество $\mathcal{P}(\zeta)$ автомата $M(\zeta)$, разрешая в качестве его элементов нейтральные дуги. Нейтральную дугу здесь следует рассматривать как сокращающее представление пары дуг, которые могут образовать гнездо только с пустым серединным элементом. Естественно также считать нейтральную дугу элементом множества $\mathcal{P}(\zeta)$.

Зная Dмножество $\mathcal{P}(\zeta)$, легко построить команды автомата:

$$\delta(\zeta) = \{(p_1, a_1, Z) \to (q_1, ZX),\ (p_2, a_2, Z) \to (q_2, \Lambda) |$$
$$((p_1, Z)a_1(q_1, X), (p_2, X)a_2(q_2, Z)) \in \mathcal{P}(\zeta)\} \cup$$
$$\{(p, a, Z) \to (q, Z) | (p, Z)a(q, Z) \in \mathcal{P}(\zeta)\}.$$

Сформулируем алгоритм постепенного построения множества $\mathcal{P}(\zeta)$, начинающий с образования дуг, инцидентных входной или заключительной вершине. Для ссылки на множества построенных дуг и их вершин будем употреблять обозначения $E(\zeta)$ и $V(\zeta)$ соответственно:

$\mathcal{P}(\zeta) := \emptyset$;

$V(\zeta) := \{[({}_0, Z_0],\ [)_0, Z_0]\}$.

Пока следующие три присваивания влекут добавление в $V(\zeta)$ новых элементов, повторять их:

$NewP := \{([({}_m, Z]\Lambda[({}_k, m],\ [)_l, m]\Lambda[)_m, Z]),\ [)_k, m]\Lambda[({}_l, m]\ |[({}_m, Z] \in V(\zeta);\ (k,l,m)$ есть альянс $\} \cup \{[({}_m, Z]a[)_m, Z] | [({}_m, Z] \in V(\zeta);\ a \in \Sigma;\ ({}_m a)_m$ есть активное гнездо $\} \cup \{[({}_m, Z]\Lambda[)_m, Z] | [({}_m, Z] \in V(\zeta);\ ({}_m\epsilon)_m$ есть активное гнездо $\}$;

$\mathcal{P}(\zeta) := \mathcal{P}(\zeta) \cup NewP$;

$V(\zeta) := V(\zeta) \cup \{P | P$ — вершина какой-либо из дуг, участвующих в элементах множества $NewP\}$.

Dграф, полученный в результате данного построения, обозначим через $D(\zeta)$. Для $s \geq 0$ и $P, Q \in V(\zeta)$ введем обозначение $\mathcal{T}(P, Q, s)$ для всех путей этого графа (они могут и не быть маршрутами) длины s с парой концов (P, Q). Пусть

$$\mathcal{T}(P, Q) = \cup_{s \geq 0} \mathcal{T}(P, Q, s).$$

Назовем *путем КСвыражения* ζ элемент следующего рекурсивно определяемого множества $Paths(\zeta)$:

$$Paths(\zeta) = Clan(\zeta) \cup \{x[(_\iota u]^k (_\iota y)_\iota [v)_\iota]^l z | x(_\iota u(_\iota y)_\iota v)_\iota z \in Paths(\zeta), \quad k, l \geq 0\}.$$

Содержательно, пути получаются из фракций нарушением "баланса"вхождений парных циклов.

Для $s \geq 0$ и $b_1, b_2 \in \mathcal{B}(Augm(\zeta))$ введем обозначение $DisBal(b_1, b_2, s)$ для следующего множества непустых участков путей из $Paths(Augm(\zeta))$:

$$DisBal(b_1, b_2, s) = \{b_1 y b_2 | s = |projection(b_1 y b_2, \mathcal{B}(Augm(\zeta)))| - 1\}.$$

Здесь значение s аналогично длине пути на графе, измеряемой в дугах. Пусть

$$DisBal(b_1, b_2) = \cup_{s \geq 0} DisBal(b_1, b_2, s).$$

Для вершины $P = (p, Z)$ обозначим состояние p через $state(P)$. Определим морфизм

$$\Phi : E(\zeta)^* \to (\{state(beg(\pi)) | \pi \in E(\zeta)\}(\Sigma \cup \{\Lambda\}))^*,$$

полагая

$$\Phi(\pi) = state(beg(\pi))\omega(\pi).$$

Здесь значение s аналогично длине пути на графе, измеряемому в дугах.

Лемма 16. Для любых $s \geq 0$ и $P, Q \in V(\zeta)$, если $\mathcal{T}(P, Q, s) \neq \emptyset$, то существует такая биекция

$$\varphi : \mathcal{T}(P, Q, s) \to DisBal(state(P), state(Q), s),$$

что для каждого пути $T \in \mathcal{T}(P, Q, s)$ выполняется равенство

$$\Phi(T) state(end(T)) = \varphi(T).$$

Доказательство. Докажем лемму индукцией по s. Рассмотрим случай $s = 0$. $\mathcal{T}(P, Q, 0) = \{P\}$ при $P = Q$ и пусто при $P \neq Q$. При $state(P) = state(Q)$ имеем $DisBal(state(P), state(Q), 0) = \{state(P)\}$, при $state(P) \neq state(Q)$ $DisBal(state(P), state(Q), s)$ пусто. Следовательно, в данном случае лемма верна.

Пусть $s > 0$. Предположим, что для любых $0 \leq s' < s$ и $P', Q' \in V(\zeta)$, удовлетворяющих неравенству $\mathcal{T}(P', Q', s') \neq \emptyset$, существует такая биекция

$$\varphi' : \mathcal{T}(P', Q', s') \to DisBal(state(P'), state(Q'), s'),$$

что для каждого пути $T \in \mathcal{T}(P', Q', s')$ выполняется равенство

$$\Phi(T) state(end(T)) = \varphi'(T).$$

Пусть $\mathcal{T}(P, Q, s) \neq \emptyset$. Тогда из построения множеств $E(\zeta)$ и $V(\zeta)$ следует возможность ровно двух случаев: $P = [(_m, Z]$ для некоторых $m \in Names(Augm(\zeta))$

и $Z \in Names(Augm(\zeta)) \cup \{Z_0\}$ или $P =]_k, m]$ для некоторых $k, m \in Names(Augm(\zeta))$. Каждая из вершин

$$[(_m, Z], \quad]_k, m]$$

может быть начальной вершиной дуг двух видов. Следовательно, $\mathcal{T}(P, Q, s)$ совпадает с одним из следующих двух объединений:

$\{[(_m, Z]\Lambda[(_k, m]T | P = [(_m, Z]$; $(\exists l \in Names(Augm(\zeta))\ (k, l, m)$ есть альянс); $T \in \mathcal{T}([(_k, m], Q, s-1)\} \cup \{[(_m, Z]a]_m, Z]T | P = [(_m, Z]$; $a \in \Sigma$; $(_m a)_m$ есть активное гнездо; $T \in \mathcal{T}(]_m, Z], Q, s-1)\} \cup \{[(_m, Z]\Lambda]_m, Z]T | P = [(_m, Z]$; $(_m \epsilon)_m$ есть активное гнездо; $T \in \mathcal{T}(]_m, Z], Q, s-1)\}$;

$\{]_k, m]\Lambda[(_l, m]T | P =]_k, m]$; (k, l, m) есть альянс; $T \in \mathcal{T}([(_l, m], Q, s-1)\} \cup \{]_k, m]\Lambda]_m, Z]T | P =]_k, m]$; $(\exists l \in Names(Augm(\zeta))\ (l, k, m)$ есть альянс); $T \in \mathcal{T}(]_m, Z], Q, s-1)\}$.

Цепочка из множества $DisBal(b_1, b_2, s)$ начинается скобкой b_1 и заканчивается скобкой b_2. Скобка b_1 может быть как открывающей, так и закрывающей. Согласно структуре фракций выражения $Augm(\zeta)$ в пути из $Paths(Augm(\zeta))$ за открывающей скобкой идет либо открывающая скобка, либо символ $a \in \Sigma \cup \{\epsilon\}$, а закрывающая скобка, отличная от $)_0$, всегда предшествует скобке, открывающей или закрывающей. Более того, указанные двухсимвольные участки путей из $Paths(Augm(\zeta))$ несут информацию об активных гнездах и альянсах, которую мы выразим в двух нижеследующих формулах, характеризующих значение множества $DisBal(state(P), state(Q), s)$ при $state(P) = (_m$ и $state(P) =)_k$ соответственно. Здесь $k, m \in Names(Augm(\zeta))$.

$\{(_m(_k y | state(P) = (_m$; $\exists l \in Names(Augm(\zeta))\ (k, l, m)$ есть альянс; $(_k y \in DisBal((_k,\ state(Q), s-1)\} \cup \{(_m a)_m y | state(P) = (_m$; $a \in \Sigma \cup \{\epsilon\}$; $(_m a)_m$ есть активное гнездо; $)_m y \in DisBal()_m, state(Q), s-1)\}$.

$\{)_k(_l y | state(P) =)_k$; $\exists m \in Names(Augm(\zeta))\ (k, l, m)$ есть альянс; $(_l y \in DisBal((_l,\ state(Q), s-1)\} \cup \{)_k)_m y | state(P) =)_k$; $\exists l \in Names(Augm(\zeta))\ (l, k, m)$ есть альянс; $)_m y \in DisBal()_m, state(Q), s-1)\}$.

Сопоставляя последние формулы с формулами для $\mathcal{T}(P, Q, s)$, выводим, что из существования биекции φ' следует, что отображение φ, определяемое соотношениями

$$\varphi(\pi T) = \Phi(\pi)\varphi'(T),\ \pi T \in \mathcal{T}(P, Q, s),$$

есть требуемая биекция $\square$

Очевидно

Следствие 2. Для любых $P, Q \in V(\zeta)$, если $\mathcal{T}(P, Q) \neq \emptyset$, то существует такая биекция

$$\varphi : \mathcal{T}(P, Q) \to DisBal(state(P), state(Q)),$$

что для каждого пути $T \in \mathcal{T}(P, Q)$ выполняется равенство

$$\Phi(T) state(end(T)) = \varphi(T)$$

$\square$

Следствие 3. $L(D(\zeta)) = L(\zeta)$.

Доказательство. По следствию 2 существует такая биекция

$$\varphi : Sentences(D(\zeta)) \to Clan(Augm(\zeta)),$$

что для каждого предложения T выполняется равенство

$$\Phi(T)state(end(T)) = \varphi(T).$$

Следовательно,

$$\omega(T) = projection(\varphi(T), \Sigma).$$

Значит,

$$L(D(\zeta)) = L(Augm(\zeta)).$$

Но КСвыражения $Augm(\zeta)$ и ζ эквивалентны $\square$

Заметим, что результирующий магазинный автомат можно до определенной степени оптимизировать механически, подобно тому, как можно механически превратить конечный автомат в эквивалентный детерминированный конечный автомат. Действительно, в главе 2 конструктивно доказано, что изъятие определенным образом команд магазинного автомата, не читающих входных символов и не стирающих в магазине, дает эквивалентный магазинный автомат. Описанное в главе 2 преобразование сохраняет свойство детерминированности.

Пример 13. КСвыражение $\zeta = ({}_1a({}_1\epsilon)_1b)_1$ задает язык $\{a^nb^n | n \geq 0\}$. $Core(\zeta, 2)$ содержит два элемента: ζ и $({}_1\epsilon)_1$. $Mark(\zeta)$ содержит

$$({}_0({}_{11}\epsilon)_{11}({}_1({}_{121}a)_{121}({}_{122}({}_1\epsilon)_1({}_{1221}b)_{1221})_{122})_1)_0 \text{ и } ({}_0({}_{21}\epsilon)_{21}({}_1\epsilon)_1)_0.$$

Итак,

$$Names(Augm(\zeta)) = \{0, 1, 11, 21, 121, 122, 1221\}.$$

Активные гнезда представляют четыре альянса:

$$(11, 1, 0),\ (121, 122, 1),\ (1, 1221, 122)\ (21, 1, 0).$$

Предоставляем читателю вычислить множество $Core(Augm(\zeta), 3)$ и убедиться, что других альянсов для данного примера не существует.

Автомат $M(\zeta)$ есть семерка

$$(K_\zeta,\ \{a, b\},\ \{Z_0\} \cup Names(Augm(\zeta)), Z_0,\ \delta_\zeta,\ ({}_0,\ \{)_0\}),$$

где $K_\zeta = \{({}_m,)_m | m \in Names(Augm(\zeta))\}$, а δ_ζ определяется следующими командами:

$(({}_0, \Lambda, Z_0) \to (({}_\iota, Z_0 0)$;
$(({}_\iota, \Lambda, 0) \to ()_\iota, 0)$ и $()_\iota, \Lambda, 0) \to (({}_1, 0)$, где $\iota \in \{11, 21\}$;
$(({}_1, \Lambda, Z) \to ()_1, Z)$ и $(({}_1, \Lambda, Z) \to (({}_{121}, Z1)$, где $Z \in \{0, 122\}$;
$(({}_{121}, a, 1) \to ()_{121}, 1)$;
$()_{121}, \Lambda, 1) \to (({}_{122}, 1)$;
$(({}_{122}, \Lambda, 1) \to (({}_1, 1\ 122)$;
$()_1, \Lambda, 122) \to (({}_{1221}, 122)$;
$(({}_{1221}, b, 122) \to ()_{1221}, 122)$;
$()_{1221}, \Lambda, 122) \to ()_{122}, \Lambda)$;
$()_{122}, \Lambda, 1) \to ()_1, \Lambda)$;
$()_1, \Lambda, 0) \to ()_0, \Lambda)$.

Составим формулу для множества успешных маршрутов построенного автомата. В данном примере можно отождествить дугу с некоторой тройкой из

$$V(\zeta) \times (\Sigma \cup \{\Lambda\}) \times V(\zeta),$$

так как граф рассматриваемого магазинного автомата не имеет кратных дуг с одинаковыми пометками. В записи маршрута будем однократно выписывать общую вершину двух последовательных дуг. Предварительно введем буквенные обозначения для элементов множества $V(\zeta)$, так как это сделает запись дуги компактней: $A = [(_0, Z_0]$, $B = [(_{11}, 0]$, $C =])_{11}, 0]$, $D = [(_1, 0]$, $E =])_1, 0]$, $F =])_0, Z_0]$, $G = [(_{121}, 1]$, $H =])_{121}, 1)$, $I = [(_{122}, 1]$, $J =])_{122}, 1]$, $K = [(_1, 122]$, $L =])_1, 122]$, $M = [(_{1221}, 122]$, $N =])_{1221}, 122]$, $P = [(_{21}, 0]$, $Q =])_{21}, 0]$.

Если положить $T_1 = A\Lambda P\Lambda Q\Lambda D$, $T_2 = A\Lambda B\Lambda C\Lambda D$, $T_3 = D\Lambda E\Lambda F$, $T_4 = GaH\Lambda I\Lambda K\Lambda G$ и $T_5 = J\Lambda L\Lambda MbN\Lambda J$, то искомая формула есть

$$\{T_1, T_2\}(\{T_3\} \cup \{T_4^i\ GaH\Lambda I\Lambda K\Lambda L\Lambda MbN\Lambda J\ T_5^i\ J\Lambda E\Lambda F | i \geq 0\}).$$

3.3. Согласующие выражения. Пусть $projection(w, \Sigma) \in \Sigma^+$ для цикла w некоторой фракции над Σ. Тогда назовем цикл w *помеченным*.

Пусть $\xi = u(_\iota x(_\iota y)_\iota z)_\iota v$ есть фракция, в которой циклы $(_\iota x$ и $z)_\iota$ оба являются помеченными. Тогда назовем фракцию ξ *вставляющей* (подразумевается вставление цепочки именованного языка внутрь цепочки того же языка, наблюдающееся в результате образования фракций $u((_\iota x)^m(_\iota y)_\iota(z)_\iota)^m v$).

Пусть ζ — КСвыражение. Если множество $Clan(\zeta)$ содержит вставляющую фракцию, то назовем выражение ζ *согласующим*, иначе *псевдосогласующим*.

Лемма 17. Если ζ — согласующее выражение, то существует такая вставляющая фракция $\xi \in Clan(\zeta)$, что $height(\xi) \leq 5$.

Доказательство. Пусть $\xi' \in Clan(\zeta)$ — вставляющая фракция. Согласно определению вставляющей фракции имеем равенство

$$\xi' = u'(_\iota x'_1 a x'_2(_\iota y')_\iota z'_1 b z'_2)_\iota v'$$

для некоторых имени $\iota \in Names(\zeta)$, цепочек

$$u', x'_1, x'_2, y', z'_1, z'_2, v' \in Alph(\zeta)^*$$

и символов $a, b \in \Sigma$. Пусть

$$\xi = u(_\iota x_1 a x_2(_\iota y)_\iota z_1 b z_2)_\iota v) \in reduction(u', (_\iota, x'_1, a, x'_2, (_\iota, y',)_\iota, z'_1, b, z'_2,)_\iota, v'),$$

где

$$s \in \{u, x_1, x_2, y, z_1, z_2, v\}$$

обозначает образ цепочки s' при данном редуцировании.

Заметим, что $projection(\xi, \mathcal{B}(\zeta))$ является скобочной системой. Следовательно, если некоторый цикл содержит выделенную в записи фракции ξ скобку $(_\iota$ или $)_\iota$, то парная ей скобка содержится либо в том же цикле, либо в парном ему цикле. Учитывая данное обстоятельство, докажем, что $h(k, \xi) \leq 5$ для любого имени $k \in Names(\zeta)$.

Заметим дополнительно, что цикл фракции ξ' отображается при данном редуцировании на цикл фракции ξ тогда и только тогда, когда он сам или парный ему цикл содержит хотя бы один из символов, сохранение которых обязательно.

Рассматривая все допустимые варианты положения скобок $(_k$ и $)_k$, $k \in Names(\zeta)$, во фракции ξ, нетрудно убедиться, что при $k \neq \iota$ максимум числа

$h(k,\xi)$ достигается в случаях, когда фракция ξ имеет одну из следующих двух форм:

$$u_1({}_k u_2({}_\iota x_{11}({}_k x_{12} a x_{21}({}_k x_{22}({}_k x_{23}({}_\iota y_1({}_k y_2)_k y_3)_\iota z_{11})_k z_{12} b z_{21})_k z_{22})_k z_{23})_\iota v_1)_k v_2,$$

$$u_1({}_k u_2({}_\iota x_{11}({}_k x_{12}({}_k x_{13} a x_{21}({}_k x_{22}({}_\iota y_1({}_k y_2)_k y_3)_\iota z_{11})_k z_{12})_k z_{13} b z_{21})_k z_{22})_\iota v_1)_k v_2,$$

а при $k = \iota$ — в следующих двух случаях:

$$\xi = u({}_\iota x_{11}({}_\iota x_{12} a x_2({}_\iota y)_\iota z_{11})_\iota z_{12} b z_2)_\iota v,$$

$$\xi = u({}_\iota x_1 a x_{21}({}_\iota x_{22}({}_\iota y)_\iota z_1 b z_{21})_\iota z_{22})_\iota v.$$

Итак, $h(k,\xi) \le 5$ для $k \in Names(\zeta)$, т.е. $height(\xi) \le 5$. Парные циклы ${}_\iota x_1 a x_2$ и $z_1 b z_2)_\iota$ являются помеченными, так как $projection({}_\iota x_1 a x_2, \Sigma)$ и $projection(z_1 b z_2)_\iota$, Σ) содержат символы a и b соответственно. Следовательно, фракция ξ является вставляющей $\square$

Лемма 18. КСвыражение ζ является псевдосогласующим тогда и только тогда, когда каждая фракция любого элемента из $Core(\zeta, 5)$ не является вставляющей.

Доказательство. Необходимость ясна из определений. Для доказательства достаточности заметим, что противное не совместимо с леммой 17 $\square$

Теорема 9. Если ζ — псевдосогласующее КСвыражение, то язык $L(\zeta)$ регулярен.

Доказательство. Граф $\mathcal{A}(\zeta)$ с множеством вершин $V(\zeta)$ и множеством дуг $E(\zeta)$ представляет конечный автомат, эквивалентный псевдосогласующему выражению ζ (если, как и для графа $D(\zeta)$, вершины $[(_0, Z_0]$ и $])_0, Z_0]$ считаются входной и заключительной соответственно).

Действительно, если ζ — псевдосогласующее КСвыражение, то для определения языка $L(\zeta)$ можно вместо $Clan(\zeta)$ использовать $Paths(\zeta)$:

$$L(\zeta) = \cup_{\xi \in Paths(\zeta)} \omega(\xi),$$

так как требование баланса парных циклов избыточно в данном случае. Заметим, что множества $Paths(\zeta)$ и $Paths(Augm(\zeta))$ определяют один и тот же язык, а

$$Paths(Augm(\zeta)) = DisBal((_0,)_0).$$

По следствию 1 существует такая биекция

$$\varphi : \mathcal{T}([(_0, Z_0],])_0, Z_0]) \to DisBal((_0,)_0),$$

что для каждого пути $T \in \mathcal{T}([(_0, Z_0],])_0, Z_0])$ выполняется равенство

$$\Phi(T) state(end(T)) = \varphi(T).$$

Последнее соотношение влечет равенство

$$\omega(\varphi(T)) = \{\omega(T)\}.$$

Но множество $\mathcal{T}([(_0, Z_0],])_0, Z_0])$ есть множество успешных путей автомата $\mathcal{A}(\zeta)$. Следовательно,

$$L(\zeta) = \{\omega(T) | T \in \mathcal{T}([(_0, Z_0],])_0, Z_0])\} = L(\mathcal{A}(\zeta))$$

$\square$

3.4. Об укорачивании КСвыражений. Начнем с примера, объясняющего идею укорачивания.

Пример 14. КСвыражение

$$\zeta = a({}_1a({}_1\epsilon)_1b)_1b({}_1a({}_1\epsilon)_1b)_1$$

задает язык

$$\{a\}\{a^nb^n|n \geq 0\}\{b\}\{a^nb^n|n \geq 0\} = \{a^mb^ma^nb^n|m \geq 1, n \geq 0\}.$$

Из определений КСвыражения и задаваемого им языка следует, что КСвыражения

$$a({}_1\epsilon)_1b({}_1a({}_1\epsilon)_1b)_1$$

и

$$a({}_1a({}_1\epsilon)_1b)_1b({}_1\epsilon)_1$$

содержат всю информацию, представленную КСвыражением ζ. Они отличаются от ζ тем, что имеют меньше вхождений одноименных гнезд, и заметно короче, чем ζ.

Далее мы займемся сокращением длины КСвыражения только за счет удаления избыточных вхождений гнезд, не пытаясь выполнять какие-либо более тонкие эквивалентные преобразования. Однако и данная скромная задача представляет интерес. Действительно, в главе 2 будет определено достаточно значимое понятие ядра бесконтекстной грамматики, задача компактного представления которого во многом сводится к данной.

КСвыражение $\emptyset$ не нуждается в укорачивании. Рассмотрим поэтому случай КСвыражения ζ, для которого

$$Trim(\zeta) \neq \emptyset.$$

Построим алгоритм укорачивания КСвыражений, которые являются суммами приведенных фракций. Такое выражение ζ не обязательно совпадает с

$$\sum_{\xi \in Trim(\zeta)} \xi,$$

так как может содержать несколько экземпляров своей приведенной фракции. (Эта неэкономность также будет преодолеваться.)

Заметим, что данный класс КСвыражений определяет все непустые бесконтекстные языки, поскольку любое КСвыражение, задающее непустой язык, эквивалентно сумме своих приведенных фракций.

Объявленный алгоритм можно интерпретировать как алгоритм укорачивания множества приведенных фракций произвольного КСвыражения. Важное приложение алгоритма состоит в том, что он указывает способ укорачивания ядра и характеристики бесконтекстной грамматики (см. п.2.2.1).

В алгоритме укорачивания очень существенным будет исследование вхождений гнезд и устранение избыточности в информации, сообщаемой о них рассматриваемыми фракциями. Метод устранения избыточности окажется более плодотворным, если считать именованным и каждый элемент множества $Trim(\zeta)$. Так и сделаем, а именно, заставим алгоритм начать с переработки множества

$$ATrim(\zeta) = \{({}_0\xi)_0|\xi \in Trim(\zeta)\},$$

считая, что

$$0 \notin Names(\zeta).$$

Начальная буква A в данном и в двух следующих обозначениях взята от слова $Augmentation$.

$$ANames(\zeta) = Names(\zeta) \cup \{0\},$$
$$AAlph(\zeta) = Alph(\zeta) \cup \{(_0,)_0\}.$$

Нижеследующее вспомогательное обозначение использует понятие "пополненных"фракций как основу:

$$Alternatives(\iota) = \{\xi | \exists (x, y, z \in AAlph(\zeta)^*)$$
$$[x(_\iota y)_\iota z \in ATrim(\zeta);\ \xi \in reduction(y)]\}$$

для

$$\iota \in ANames(\zeta).$$

Сформируем укорочение

$$Squeeze(\zeta)$$

КСвыражения ζ как сумму, возможно, вырожденную — из одного слагаемого. Может случиться, что слагаемое КСвыражения $Squeeze(\zeta)$ не будет удовлетворять определению фракции, т.е. в нем к невырожденным суммам будет применяться операция умножения или именования.

Приближения искомого КСвыражения будут представлены цепочкой

$$Sqz.$$

Образуем ее, руководствуясь следующим порядком употребляемых имен гнезд. Начнем с имени 0, а остальные имена будем перебирать в порядке их первого появления в последовательных значениях величины Sqz. При этом порядок перебора имен, появившихся одновременно, не является существенным.

Множество уже привлеченных имен обозначим через

$$Building,$$

очередных добавляемых к рассмотрению —

$$Storey.$$

Ясно, что при первоначально пустом множестве $Building$ добавлений к нему будет не больше числа $|ANames(\zeta)|$.

Гнезда в формируемых слагаемых будут до поры до времени изображаться цепочками вида

$$(_\iota)_\iota, \quad \iota \in ANames(\zeta).$$

(Такую цепочку будем называть *ι-заглушкой*, или, собирательно, *заглушкой*. В связи с этим для описания "полуфабриката"КСвыражения $Squeeze(\zeta)$ полезна функция, отображающая фракцию в цепочку заглушек и символов языка; для произвольной фракции ξ над алфавитом Σ эта функция определяется следующим образом:

$$ground(\xi) = \begin{cases} \xi,\ \xi \in \Sigma \cup \{\epsilon\}, \\ (_\iota)_\iota,\ \xi = (_\iota u)_\iota \text{ для } \iota \in ANames(\zeta),\ u \in Alph(\zeta)^*, \\ ground(\xi_1)ground(\xi_2),\ \xi = \xi_1\xi_2 \text{ для фракций } \xi_1, \xi_2. \end{cases}$$

Устранение заглушек будет осуществляться с помощью описываемой далее процедуры

$$Replacement(\iota).$$

Устранение происходит поэтапно, при все более подробной их "расшифровке", раскрытии. Раскрывающий материал предоставляется на последнем этапе — множеством $Alternatives(\iota)$, вернее, самыми короткими его элементами, на промежуточных этапах — величиной

$$Embeddings(\iota).$$

Ее значением является множество

$$\{u|\exists[\nu \in ANames(\zeta) - Building - Storey,\ x, y, z \in Alph(\zeta)^*]$$

$$[\xi = x(_\nu y)_\nu z \in Alternatives(\iota),\ u = ground(\xi)]\}$$

цепочек, вовлекающих еще не использовавшиеся в Sqz имена. Эти имена составляют множество

$$New(\iota) = \{\nu \notin Building \cup Storey|\exists[x, y \in Alph(\zeta)^*]$$

$$x(_\nu)_\nu y \in Embeddings(\iota)\}.$$

Алгоритм 3. Укорачивание КСвыражения.

Вход. КСвыражение

$$\zeta \neq \emptyset,$$

являющееся суммой приведенных фракций.

Выход. КСвыражение

$$Squeeze(\zeta),$$

определяемое конструкцией алгоритма.

Шаг 1. $Sqz := (_0)_0$;
$Building := \emptyset$;
$Storey := \{0\}$.

Шаг 2. Пока $Storey \neq \emptyset$, выполнять следующие действия 2.1–2.2.

2.1. Для каждого имени $\iota \in Storey$ вычислить множества $New(\iota)$ и $Embeddings(\iota)$ и выполнить процедуру $Replacement(\iota)$.

2.2. $Building := Building \cup Storey$;
Вычислить объединение

$$\cup_{\iota \in Storey} New(\iota)$$

и сделать его новым значением величины $Storey$.

Шаг 3. Для каждого имени $\iota \in Names(\zeta)$ заменить каждую ιзаглушку, еще присутствующую в цепочке Sqz, гнездом $(_\iota \xi)_\iota$, где $\xi \in Alternatives(\iota)$ есть какая-либо из альтернатив, имеющих минимальную длину.

Шаг 4. Стереть "искусственные"скобки $(_0,\ )_0$. Положить результирующее КСвыражение равным значению величины Sqz.

Процедура $Replacement$ внедряет в цепочку Sqz вместо заглушек с заданным именем (возможно, не всех) некоторые их расшифровки. Не раскрытые заглушки обрабатываются шагом 3 алгоритма 3. Для этого используются кратчайшие из альтернатив, определяющих одноименные гнезда КСвыражения ζ.

Число раскрываемых процедурой ιзаглушек есть минимальное из числа их вхождений в Sqz и $|Embeddings(\iota)|$. Если последние два числа равны, то каждому вхождению заглушки отвечает один элемент из $Embeddings(\iota)$. При избытке материала, подлежащего внедрению, какой-либо один экземпляр заглушки заменяется именованной суммой (например, при наличии нескольких фракций в $ATrim(\zeta)$ внутрь исходной 0заглушки попадет сумма). Таким образом, используется весь этот материал. При нехватке материала обработка "лишних"заглушек откладывается, чтобы не увеличивать длину цепочки Sqz.

Алгоритм 4. $Replacement$

Вход. Имя $\iota \in ANames(\zeta)$; текущее значение величины Sqz.

Выход. Преобразованное значение Sqz.

Вспомогательные обозначения. Число m_ι вхождений ιзаглушки в Sqz;

$$m = min(|Embeddings(\iota)|,\ m_\iota);$$

$$Individuals \subseteq Embeddings(\iota),\ Group \subseteq Embeddings(\iota)$$

есть одна любая из пар множеств, удовлетворяющих соотношениям

$$|Individuals| = m - 1,$$

$$Group = Embeddings(\iota) - Individuals.$$

Шаг 1. Выбрать какие-либо $m - 1$ вхождений в Sqz ιзаглушки, например, первые при переборе вхождений слева направо. Определить одну из возможных биекций выбранного множества на множество $Individuals$. Каждое из выбранных вхождений заменить ιгнездом

$$({}_\iota u)_\iota,$$

где

$$u \in Indiviiduals$$

есть образ данного вхождения ιзаглушки.

Шаг 2. Одно из незаменявшихся вхождений ιзаглушки в Sqz (таких может быть больше одного при $|Embeddings(\iota)| < m_\iota$) заменить ιгнездом, заключающим сумму элементов множества $Group$:

$$({}_\iota \sum_{u \in Group} u)_\iota.$$

Пусть ι — некоторое имя, ξ — ιгнездо. Определим рекурсивно понятие *среза* гнезда ξ и числовую характеристику среза, называемую его *статусом*:

1) заглушка $({}_\iota)_\iota$ является срезом гнезда ξ, имеющим нулевой статус;

2) если

$$\xi = ({}_\iota u)_\iota$$

и для некоторого $l \geq 0$

$$u = v_0 \xi_1 v_1 \ldots \xi_l v_l,$$

где цепочка v_j, $0 \leq j \leq l$, не содержит именованных скобок, а ξ_i, $1 \leq i \leq l$, есть гнездо, имеющее срез u_i со статусом s_i, то цепочка

$$({}_\iota v_0 u_1 v_1 \ldots u_l v_l)_\iota$$

является срезом гнезда ξ, имеющим статус

$$1 + max\{s_i | 1 \leq i \leq l\}.$$

Заметим, что при работе шагов 2–3 алгоритма 3 цепочка Sqz представляет срезы 0гнезда КСвыражения $({}_0Squeeze(\zeta))_0$.

Заметим, что для $s \geq 1$ срез со статусом s гнезда

$$({}_\iota u)_\iota$$

получается подстановкой в

$$({}_\iota ground(u))_\iota$$

вместо заглушек некоторых срезов со статусом, не превышающим $s-1$. В частности,

$$({}_\iota ground(u))_\iota$$

является срезом гнезда $({}_\iota u)_\iota$, имеющим статус 1.

Пусть для любого КСвыражения ξ

$$Nests(\xi)$$

есть обозначение множества всех гнезд, встречающихся во фракциях из $Clan(\xi)$.

Пусть ξ есть КСвыражение, s — неотрицательное целое. Обозначим через $Sections(\xi, s)$ множество всех срезов со статусом s гнезд, участвующих в $Nests(\xi)$.

Вернемся к алгоритму 3. В следующей теореме, как и раньше, предполагаем, что

$$0 \notin Names(\zeta).$$

Теорема 10. Для любого неотрицательного целого s справедливо равенство

$$Sections(({}_0\zeta)_0, s) = Sections(({}_0Squeeze(\zeta))_0, s).$$

Доказательство. Индукцией по s.

При $s = 0$ рассматриваемое множество срезов является множеством заглушек. По определению клана множество $Clan(({}_0\zeta)_0)$ определяется через

$$Trim(({}_0\zeta)_0) = ATrim(\zeta).$$

Из конструкции шагов 1, 2 алгоритма 3 следует, что КСвыражение $({}_0Squeeze(\zeta))_0$ и его клан представляют все заглушки, представленные в $ATrim(\zeta)$, и только их. Следовательно, теорема в данном случае верна.

Пусть теперь $s > 0$. Тогда каждый рассматриваемый срез представлен для некоторого целого $l \geq 0$ цепочкой вида

$$({}_\iota u)_\iota = ({}_\iota v_0({}_{\iota_1} u_1)_{\iota_1} v_1 \ldots ({}_{\iota_l} u_l)_{\iota_l} v_l)_\iota,$$

где

$$ground(u) = v_0({}_{\iota_1})_{\iota_1} v_1 \ldots ({}_{\iota_l})_{\iota_l} v_l.$$

Из конструкции алгоритма (см. процедуру *Replacement*) следует, что для любого имени $\iota \in ANames(\zeta)$ множества

$$\{ground(\xi) | ({}_\iota \xi)_\iota \in Nests(({}_0\zeta)_0)\}$$

и

$$\{ground(\xi)|(_\iota\xi)_\iota \in Nests((_0Squeeze(\zeta))_0)\}$$

совпадают каждое с $Embeddings(\iota)$, т.е. равны между собой.

Таким образом, каждое из множеств

$$Sections((_0\zeta)_0, s) \text{ и } Sections((_0Squeeze(\zeta))_0, s)$$

получается (ср. замечание к определению среза) из одного и того же множества

$$\cup_{\iota \in ANames(\zeta)}[\{(_\iota\}Embedding(\iota)\{)_\iota\}]$$

всевозможными подстановками вместо заглушек цепочек, входящих соответственно в

$$\cup_{0 \le j < s} Sections((_0\zeta)_0, j)$$

и

$$\cup_{0 \le j < s} Sections((_0Squeeze(\zeta))_0, j).$$

По предположению индукции

$$Sections((_0\zeta)_0, j) = Sections((_0Squeeze(\zeta))_0, j)$$

для $0 \le j < s$. Значит, верно и равенство

$$Sections((_0\zeta)_0, s) = Sections((_0Squeeze(\zeta))_0, s)$$

□

Следствие 4. $Nests((_0\zeta)_0) = Nests((_0Squeeze(\zeta))_0)$.

Доказательство. Заметим, что каждое гнездо ξ является своим срезом, т.е. принадлежит множеству $Sections(\xi, s)$ для некоторого целого $s \ge 1$. Так как каждый элемент кланов интересующих нас КСвыражений является 0ғнездом, неравенство

$$Nests((_0\zeta)_0) \ne Nests((_0Squeeze(\zeta))_0)$$

противоречило бы теореме 10 □

Следствие 5. $L((_0\zeta)_0) = L((_0Squeeze(\zeta))_0)$.

Доказательство. Согласно следствию 4 верно равенство

$$Clan((_0\zeta)_0) = Clan((_0Squeeze(\zeta))_0).$$

Значит, по определению задаваемого КСвыражением языка имеем равенство рассматриваемых языков □

Теорема 11. $|Squeeze(\zeta)| \le |\zeta|$.

Доказательство. Положим

$$n_\zeta = |(_0\zeta)_0|$$

и убедимся, что в каждый момент работы алгоритма 3 выполняется неравенство

$$|Sqz| \le n_\zeta.$$

Оно очевидно для шага 1. Действительно, запись выражения ζ представляет собой непустую цепочку, поэтому на шаге 1 имеем

$$|Sqz| = |(_0)_0| < |(_0\zeta)_0| = n_\zeta.$$

На шаге 2 значение величины Sqz изменяется процедурой $Replacement$. Процедура заменяет заглушки одноименными срезами статуса 1 гнезд КСвыражения $({}_0\zeta)_0$.

Из определения среза видно, что срез образован сцеплением некоторых участков (возможно, разъединенных) соответствующего гнезда. Будем говорить, что срез *покрывает* этот набор участков гнезда (или КСвыражения, объемлющего гнездо). Теперь отметим очень важное в доказательстве данной теоремы обстоятельство, вытекающее из закона взаиморасположения гнезд: различные срезы единичного статуса покрывают в КСвыражении наборы участков, элементы которых попарно не перекрываются. (Это верно и для одинаковых на вид срезов статуса 1, относящихся к различным гнездам одного и того же КСвыражения. Но нас интересуют различающиеся срезы, так как только они собираются в множествах $Embeddings$.) Таким образом, гнездо можно "выложить", как мозаику, всеми срезами единичного статуса, которые извлекаются из его записи.

Процедура $Replacement$ использует все текстуально различные срезы статуса 1. Каждый из них используется для замены заглушки ровно один раз. Следовательно, вся совокупность цепочек, внедряемых в Sqz процедурой $Replacement$ не может превзойти по длине КСвыражения $({}_0\zeta)_0$.

По завершении шага 2 в Sqz еще могут присутствовать заглушки. Ясно, что так может быть в том и только том случае, когда какие-либо различные гнезда дают одинаковые срезы со статусом 1, т.е. когда в исходном выражении дублируется некоторая информация. Чтобы остаться в пределах рассматриваемого формализма — КСвыражения, необходимо заменить оставшиеся заглушки какими-нибудь из одноименных с ними гнезд. В целях укорочения КСвыражения следует выбрать самые короткие из возможных замен. Что и делается.

Шаг 4, исключающий из рассмотрения внешние скобки (с именем 0), не влияет на соотношение длин исходного и результирующего КСвыражений $\square$

Выявим временные характеристики алгоритма 3, опираясь на соотношение (см. доказательство теоремы 11)

$$|Sqz| \le n_\zeta.$$

Ясно, что время работы шагов 1 и 4 не зависит от входного выражения и пренебрежимо мало, а время работы шагов 2 и 3 по меньшей мере линейно зависит от n_ζ. Действительно, для построения всех используемых алгоритмом вспомогательных объектов требуется линейный просмотр исходного выражения или не более длинной цепочки Sqz.

Так как гнездо и заглушка обязательно имеют в своем составе два символа: левую и правую скобки, число гнезд и заглушек в цепочке Sqz не превышает

$$\frac{|Sqz|}{2} \le \frac{n_\zeta}{2}.$$

Текст шага 3 алгоритма 3 очень краток, и кажется, что время работы шага 3 сравнительно легко оценить. Начнем с него, подготавливая соображения, полезные и в будущем.

В случае простейшего представления цепочки Sqz шаг 3 должен затрачивать на поиск в ней последовательных вхождений заглушек время, пропорциональное ее длине. На обработку всех найденных заглушек потребуется не более половины этой длины (ср. замечание выше о числе заглушек в Sqz).

При оценке времени работы шага 3 естественно учесть время t_ι поиска в множествах

$$Alternatives(\iota), \quad \iota \in ANames(\zeta),$$

элементов минимальной длины.

Подытоживая, получаем верхнюю оценку

$$k_{31} \cdot n_\zeta + k_{32} \cdot |ANames(\zeta)| \cdot (\frac{n_\zeta}{2} + max\{t_\iota | \iota \in ANames(\zeta)\}),$$

где k_{31} и k_{32} — некоторые коэффициенты, на значения которых могут повлиять лишь способ представления обрабатываемых данных и удачность приемов обработки.

Время t_ι в худшем случае пропорционально величине $\frac{n_\zeta}{2}$, так как число элементов в $Alternatives(\iota)$ не превосходит числа гнезд в ${}_0(\zeta)_0$. Заменяя t_ι на $k_{33} \cdot \frac{n_\zeta}{2}$, преобразуем формулу оценки:

$$(k_{31} + k_{32}\frac{1 + k_{33}}{2}|ANames(\zeta)|) \cdot n_\zeta.$$

Вводя обозначение k_3 для не зависящего от характеристик КСвыражения ζ коэффициента $k_{32}\frac{1+k_{33}}{2}$, имеем оценку

$$(k_{31} + k_3 \cdot |ANames(\zeta)|) \cdot n_\zeta$$

времени работы шага 3.

Для оценки времени работы шага 2 алгоритма 3 существенны временная оценка процедуры $Replacement$ и число ее применений. Последнее равно $|ANames(\zeta)|$. В самом деле, правила изменения алгоритмом значений величин $Building$ и $Storey$ обеспечивают однократное попадание в $Storey$ каждого имени $\iota \in ANames(\zeta)$ и, следовательно (см. "заголовок цикла", представленного действием 2.1), однократное применение процедуры $Replacement$ для каждого имени $\iota \in ANames(\zeta)$.

Сама процедура $Replacement$ движется одновременно по Sqz и по $Embeddings(\iota)$, обрабатывая $m \leq \frac{n_\zeta}{2}$ (см. вспомогательные обозначения процедуры) вхождений заглушек. Время движения можно считать пропорциональным n_ζ. Время на обработку вхождения также можно считать пропорциональным n_ζ. Действительно, обработка — это вставка между скобками заглушки заготовленной ее расшифровки. Самое примитивное осуществление вставки потребует передвижения двух цепочек (следующей в Sqz за вхождением заглушки и цепочки-расшифровки), длина каждой из которых меньше n_ζ.

Итак, время работы шага 2 не превосходит

$$k_2 \cdot |ANames(\zeta)| \cdot n_\zeta^2,$$

где k_2 — некоторый не зависящий от характеристик ζ коэффициент.

Следовательно, время работы алгоритма 3 в целом можно считать пропорциональным произведению квадрата длины исходного КСвыражения на число имен его гнезд.

Глава 2

ОБ ОПТИМИЗАЦИИ РАСПОЗНАВАНИЯ И СИНТАКСИЧЕСКОГО АНАЛИЗА БЕСКОНТЕКСТНЫХ ЯЗЫКОВ

Глава посвящена преобразованиям языковых описаний, направленным на решение вопросов, интересных с точки зрения приложений теории бесконтекстных языков. Параграф 1 затрагивает вопрос повышения эффективности детерминированного магазинного автомата. Более эффективным полагается автомат, выполняющий меньше таких тактов, которые не читают входных символов. Для магазинного автомата, каждый такт которого, не стирающий в магазине, читает входной символ, вводится термин "автомат, наполняющий магазин в реальное время". Конструктивно доказывается, что произвольный детерминированный магазинный автомат эквивалентен детерминированному же магазинному автомату, наполняющему магазин в реальное время.

В параграфе 2 LR(k)-метод обобщается на случай произвольных бесконтекстных грамматик. Обобщенный анализатор представляет собой преобразователь с магазинной памятью. Согласно первой главе его можно интерпретировать как D-граф "с выходами". Предлагаются приемы оптимизации анализатора, имеющие достаточно очевидную D-графическую интерпретацию. В случае LR(k)-грамматик это есть и приемы уменьшения LR(k)-анализаторов. Пределы уменьшения диктуются требованием сохранения в ядре уменьшенного анализатора некоторого минимума информации о ядре исходного.

В параграфе 3 даются оригинальные алгоритмы преобразования конечного автомата в регулярное выражение и D-графа в КС-выражение. На их примере показывается, что причиной успеха методов, расщепляющих языковые описания для переработки, является определенное подчинение частей этих описаний; упомянутое подчинение удобно выразить в терминах сильно связных компонент графов. Указывается, что в силу установленной взаимосвязи грамматик, автоматов и D-графов, техника рассмотренных алгоритмов может быть перенесена, например, на случай преобразования бесконтекстной грамматики в анализатор.

§1. Магазинные автоматы, наполняющие магазин в реальное время

Одна из проблем, связанных с бесконтекстными языками, состоит в построении такого магазинного автомата, который допускает язык с наименьшей затратой времени. С точки зрения приложений особенно важно добиваться эффективности детерминированных магазинных автоматов (ДМА). ДМА эффективен в том смысле, что имитирующий его алгоритм работает без возвратов. Однако для ДМА допускаются такты, которые не читают входных символов, перерабатывая, возможно, содержимое магазина. ДМА, действующий в реальное время (ДМАРВ), можно считать идеалом: ему требуется ровно n тактов, чтобы допустить цепочку длины n. Некоторое исследование ДМАРВ проведено еще в [Ginsburg -Greibach

66], где они не имеют какого-либо специального названия; данный термин заимствован из [Романовский 86]. Известно [Ginsburg - Greibach 66], что ДМАРВ составляют собственный подкласс класса ДМА.

Здесь доказывается (конструктивно), что произвольный ДМА может быть преобразован в ДМА, каждый такт которого, не уменьшающий содержимое магазина, читает входной символ. Тем самым очерчен предел, достижимый при таком преобразовании ДМА, которое приближает его к указанному выше идеалу.

Рассматриваемый здесь алгоритм отличается от аналогичного алгоритма работы [Кузнецова -Ожиганов -Сагинтаева -Станевичене 90] способом устранения зацикливающих конфигураций (последний термин см., например, в [Aho -Ullman 72]), позволившим упростить как формулировку алгоритма, так и его обоснование.

В разделе 1.1 определяется обобщение графа магазинного автомата. В этом обобщении разрешаются, дополнительно к дугам ранее определенного вида, нейтральные дуги и дуги, накапливающие в магазине больше одного символа. В следующих разделах используются обобщенные графы магазинных автоматов.

В разделе 1.2 формулируется определение зацикливающей конфигурации магазинного автомата и рассматривается алгоритм устранения зацикливающих конфигураций ДМА. По существу, здесь построено новое доказательство однозначности детерминированного языка.

В разделе 1.3 дается определение магазинного автомата, наполняющего магазин в реальное время (АНРВ), и с помощью результатов предыдущего раздела доказывается, что каждый детерминированный язык допускается некоторым детерминированным АНРВ (ДАНРВ).

1.1. Обобщение графа магазинного автомата. Определим видоизменение графа магазинного автомата, которое не подпадает под определение Dғрафа, но в терминах которого удобно описать желаемое преобразование магазинного автомата. Об этом графе можно сказать, что он является сжатием графа магазинного автомата: некоторые маршруты, длина которых больше единицы, в результате сжатия становятся дугами. Пометки дугєжатий имеют обычный вид, но вид зарядов усложняется.

Для построения интересующего нас графа по магазинному автомату не обязательно преобразовывать магазинный автомат в совершенный; можно ослабить требования к следу команды, как сказано ниже.

В данном параграфе, если не оговорено особо, полагаем, что след каждой команды магазинного автомата

$$M = (K,\ \Sigma,\ \Gamma,\ Z_0,\ \delta,\ p_0,\ F)$$

имеет один из видов: $-Z$, $Z \in \Gamma$, или $+\gamma$, $\gamma \in \Gamma^*$. Таким образом, запрещаются только следы вида $-Z+\gamma$, где $Z \in \Gamma$, $\gamma \in \Gamma^+$. В остальном магазинный автомат произволен. Рассматриваемые далее преобразования магазинных автоматов не нарушают указанных ограничений на вид следа команды.

Термин "накапливающая", введенный в главе 1, будем теперь применять и к команде со следом $+\gamma$, где $|\gamma| > 1$.

Сформулируем следующее простое наблюдение о соответствии команд совершенного магазинного автомата его дугам; оно помогает понять конструкцию определяемого далее графа $ExtG(M)$ и его связь с графом соответствующего автомату M совершенного магазинного автомата.

Пусть

$$\theta = (p, a, Z) \to (q, \gamma)$$

есть команда совершенного магазинного автомата, участвующая в его успешных вычислениях. Тогда его граф имеет дугу с начальной вершиной (p, Z), конечной вершиной (q, X) для некоторого магазинного символа X, пометкой a и зарядом $\alpha(\theta)$. Обозначим множество таких дуг через $edges(\theta)$.

Пусть θ — команда магазинного автомата M, не укорачивающая содержимого магазина (формально, $\alpha(\theta) \notin \{-\}\Gamma$). Из конструкции алгоритма совершенствования (см. алгоритм 1.1) ясно, что он сопоставляет команде θ единственную упорядоченную последовательность команд

$$steps(\theta) = \theta_1 \dots \theta_k,\ k \geq 1,$$

которая равносильна ей по действию на состояние автомата, магазин и входную цепочку, одним словом, на конфигурацию.

Из уникальности вводимых при этом алгоритмом 1.1 состояний и магазинных символов и из отмеченного выше наблюдения следует, что $|edges(\theta_i)| = 1$ для $1 \leq i \leq k$.

Обозначим через $G(M)$ граф совершенного магазинного автомата, получаемого применением к M алгоритма 1.1.

Пусть теперь θ — произвольная команда автомата M. Сопоставим ей множество $\mathbf{T}(\theta)$ маршрутов Dграфа $G(M)$ следующим образом:

$$\mathbf{T}(\theta) = \begin{cases} edges(\theta),\ \alpha(\theta) \in \{-\}\Gamma, \\ \{\pi_1 \dots \pi_k\},\ \alpha(\theta) \notin \{-\}\Gamma,\ steps(\theta) = \theta_1 \dots \theta_k, \\ \qquad \{\pi_i\} = edges(\theta_i) \text{ для } 1 \leq i \leq k. \end{cases}$$

Определение множества $\mathbf{T}(\theta)$ показывает осуществимость построения графа $ExtG(M)$ с множеством вершин

$$V(M) = \{beg(T),\ end(T) \mid \exists \theta \in \delta\ T \in \mathbf{T}(\theta)\}$$

и множеством дуг

$$E(M) = \{beg(T)\frac{\omega(T)}{\mu(T)}end(T) \mid \exists \theta \in \delta\ T \in \mathbf{T}(\theta)\}.$$

Элементы P, Q, a, w названного дугой объекта $P\frac{a}{w}Q$ будем называть соответственно начальной вершиной, конечной вершиной, пометкой и зарядом.

Назовем входной вершиной графа $ExtG(M)$ вершину $P_0(M) = (p_0, Z_0)$, заключительной — каждую вершину множества $Fin(M) = \{(p, Z) \mid p \in F, Z \in \Gamma\} \cap V(M)$. В целом $ExtG(M)$ определяется восьмеркой

$$(K,\ \Sigma,\ \Gamma,\ Z_0,\ V(M),\ E(M),\ p_0,\ Fin(M)).$$

Отметим, что описываемые в данном параграфе преобразования обобщенных графов могут привести к появлению заключительных вершин, состояния которых не являются заключительными состояниями исходного магазинного автомата.

Определенные в параграфе 1.1 понятия вычисления, маршрута, предложения и пр. без труда переносятся на случай определенных здесь графов, но мы не пытаемся определить здесь аналог Dмножества графа совершенного магазинного автомата. Используем обозначение $Sentences(ExtG(M))$ для множества всех предложений и определим язык $L(ExtG(M))$, задаваемый графом $ExtG(M)$, формулой

$$L(ExtG(M)) = \{\omega(T) \mid T \in Sentences(ExtG(M))\}.$$

Из конструкции графа $ExtG(M)$ следует, что он эквивалентен графу $G(M)$ и, значит, самому автомату M.

Далее вместо магазинного автомата M рассматривается отвечающий ему граф $ExtG(M)$. Дуги и маршруты относятся теперь к графу $ExtG(M)$, а не к $G(M)$. По аналогии с названиями команд введем названия *нейтральный, накапливающий, стирающий* для маршрута соответственно с пустым зарядом, с зарядом $+\gamma$, с зарядом $-\gamma$, где γ — непустая цепочка магазинных символов.

Нам потребуется строить магазинный автомат по графу, полученному некоторыми преобразованиями графа $ExtG(M)$. Тогда будет полезным следующее обозначение.

Пусть

$$\pi = (p, Z)\frac{a}{w}(q, X) \in E(M).$$

Обозначим через

$$rule(\pi)$$

команду

$$\theta = (p, a, Z) \to (q, \gamma) \in \delta,$$

где $\gamma \in \Gamma^*$ определяется из равенства

$$\mu(\theta) = \mu(\pi) = w.$$

Из определения заряда команды следует, что заряд w и символ Z однозначно определяют цепочку γ. Следовательно, $rule$ — это функция на множестве $E(M)$ дуг графа $ExtG(M)$. Доопределим функцию $rule$ на множестве маршрутов графа $ExtG(M)$ по правилу

$$rule(T) = rule(\pi_1) \dots rule(\pi_m),$$

$T = \pi_1 \dots \pi_m$ — маршрут, $m \geq 0$, π_i — дуга для $i = 1, \dots, m$.

1.2. Зацикливающие конфигурации. Дугу с пустой пометкой, короче, нечитающую, будем обозначать буквой ϵ.

Пусть (p, x, γ) — конфигурация магазинного автомата M. Пусть для любого $i \geq 1$ существует конфигурация $(p_i,\ x_i,\ \gamma_i)$, такая, что

$$\mid \gamma_i \mid \geq \mid \gamma \mid$$

и

$$(p, x, \gamma) \models^i (p_i, x, \gamma_i).$$

Тогда назовем конфигурацию $(p,\ x,\ \gamma)$ *зацикливающей конфигурацией* автомата M.

Лемма 1. Пусть T_0T — маршрут ДМА, T_0 — нейтральный или накапливающий цикл с пустой пометкой, $\mid T \mid \leq \mid T_0 \mid$. Тогда $T_0 = TT_1$ для некоторого маршрута T_1.

Доказательство. Применим индукцию по длине участка T. При $\mid T \mid = 0$ имеем $T_0 = TT_0$, так как вершина пустого маршрута T является конечной вершиной цикла T_0 и совпадает с его начальной вершиной.

Предположим, что лемма верна, если $\mid T \mid \leq m$, где $0 \leq m < \mid T_0 \mid$, и рассмотрим случай такого маршрута T, что $\mid T \mid = m + 1$. Представим T в виде $T = T_2\pi$, где π — дуга. По предположению индукции T_0 имеет вид $T_0 = T_2\epsilon T_3$ для некоторых дуги ϵ и маршрута T_3. Вследствие детерминированности автомата и равенства $\omega(\epsilon) = \epsilon$ имеем для некоторой команды θ равенства

$$\theta = rule(\pi) = rule(\epsilon)$$

и, следовательно,

$$\alpha(\theta) = \mu(\pi) = \mu(\epsilon).$$

Если команда θ не является стирающей, то она определяет единственную дугу, т.е. $\pi = \epsilon$. Допустим, что θ — стирающая команда. Тогда $\alpha(\theta) = -Z$ для некоторого $Z \in \Gamma$. Стирающая команда может определять дуги, различающиеся магазинными символами своих конечных вершин. Убедимся, что рассматриваемые дуги π и ϵ не имеют такого различия. Маршрут $T_2\epsilon$ является нейтральным или накапливающим как начальный участок нейтрального или накапливающего маршрута T_0. Следовательно, $ecol(T_2) = +\gamma XZ$, а $ecol(T_2\epsilon) = +\gamma X$ для некоторых $\gamma \in \Gamma^*$ и $X \in \Gamma$. Но тогда из равенства $\mu(\pi) = \mu(\epsilon)$ вытекает, что

$$ecol(T_2\pi) = \mu(+\gamma XZ - Z) = +\gamma X.$$

Отсюда $\pi = \epsilon$ $\square$

Обозначим через $\Pi(M)$ множество маршрутов без повторяющихся дуг:

$$\Pi(M) = \{\pi_1 \dots \pi_m \mid m \geq 0,\ \pi_i \in E(M)$$

$$\text{для } i = 1, \dots, m,\ \pi_i \neq \pi_j \text{ при } i \neq j\}.$$

Из конечности множества дуг магазинного автомата следует, что для любого магазинного автомата M множество $\Pi(M)$ конечно.

Введем обозначение $C_+(M)$ для множества циклов

$$\{T \in \Pi(M) \mid T \text{ — нейтральный или накапливающий цикл, } \omega(T) = \Lambda\}.$$

Лемма 2. ДМА M имеет зацикливающую конфигурацию тогда и только тогда, когда $C_+(M) \neq \emptyset$.

Доказательство. Пусть (p, x, γ) — зацикливающая конфигурация автомата M. По определению зацикливающей конфигурации для любого $i \geq 1$ существует конфигурация (p_i, x, γ_i), такая, что $\mid \gamma_i \mid \geq \mid \gamma \mid$ и $(p, x, \gamma) \models^i (p_i, x, \gamma_i)$. Из детерминированности автомата M следуют соотношения

$$(p, x, \gamma) \models (p_1, x, \gamma_1) \models (p_2, x, \gamma_2) \models \dots$$

Рассмотрим начало данной последовательности, имеющее $|V(M)|+1$ членов. Ему отвечает маршрут с пустой пометкой, среди вершин которого есть хотя бы две одинаковые. Следовательно, маршрут этот содержит цикл, который, как и весь маршрут, имеет пустую пометку. Из неравенств $|\gamma_i| \geq |\gamma|$, где i любое, следует, что это нейтральный или накапливающий цикл. Отсюда выводим, что $C_+(M) \neq \emptyset$.

Пусть теперь $C_+(M) \neq \emptyset$. Тогда существует предложение T, которое содержит нейтральный или накапливающий цикл с пустой пометкой. Представим T в виде

$T = T_0T_1T_2T_3$, где T_1 — нейтральный или накапливающий цикл, $\omega(T_1T_2) = \Lambda$ (участок T_2 может быть пустым), T_2T_3 не содержит нейтрального или накапливающего цикла с пустой пометкой. Ввиду леммы 1 $T_1 = T_2T_3T_4$ для некоторого маршрута T_4. Следовательно, маршрут T_2T_3 является нейтральным или накапливающим как начальный участок нейтрального или накапливающего маршрута T_1. Это означает, что никакая подцепочка зарядов $\mu(T_1)$ и $\mu(T_2T_3)$ не является парной какой-либо подцепочке заряда $\mu(T_0)$. Но тогда для любого $k \geq 0$ $T_0T_1^kT_2T_3$ есть предложение и, таким образом, M имеет зацикливающую конфигурацию □

Пусть

$$\mathcal{G} = (K, \Sigma, \Gamma, Z_0, V, E, p_0, F)$$

есть граф вида, определенного в начале первого раздела, $f \notin K$. Тогда обозначим через

$$Automaton(\mathcal{G})$$

магазинный автомат $(K \cup \{f\},\ \Sigma,\ \Gamma,\ Z_0,\ \{rule(\pi) \mid \pi \in E\} \cup \{(p,\ \Lambda,\ Z) \to (f,\ Z) \mid (p,\ Z) \in Fin,\ \exists X \in \Gamma\ (p, X) \in V - Fin\},\ p_0,\ \{p \in K \mid \exists Z \in \Gamma\ (p, Z) \in Fin;\ \forall X \in \Gamma\ (p, X) \notin V - Fin\} \cup \{f\})$. Заметим, что

$$ExtG(Automaton(ExtG(M))) = ExtG(M).$$

Следующий алгоритм фактически устраняет зацикливающие конфигурации успешных вычислений ДМА.

Алгоритм 1. Устранение нечитающих концевых циклов.

Вход. ДМА $M = (K, \Sigma, \Gamma, Z_0, \delta, p_0, F)$.

Выход. $\mathcal{G} = (K, \Sigma, \Gamma, Z_0, V, E, p_0, Fin)$.

Метод. $E := \{\pi \in E(M) \mid$ любой $T \in C_+(M)$ не содержит дуги $\pi\}$;
$V := \{P \in V(M) \mid \exists \pi \in E :$ вершина P инцидентна дуге $\pi\} \cup \{P_0(M)\}$;
$Fin := V \cap [Fin(M) \cup \{P \mid P$ является вершиной некоторого цикла из $C_+(M)\}]$.

В следующей теореме используются обозначения алгоритма 1.

Теорема 1. Если M — ДМА, то:
1) $M' = Automaton(\mathcal{G})$ является детерминированным;
2) $L(M') = L(M)$; 3) M' не имеет зацикливающих конфигураций.

Доказательство. 1. Заметим, что алгоритм 1 дает значение $E \subseteq E(M)$. Но в подмножестве множества дуг детерминированного магазинного автомата свойство детерминированности сохраняется. Рассмотрим нечитающие нейтральные дуги, которые может добавить операция $Automaton$. Начальная вершина P такой дуги в исходном автомате является начальной вершиной некоторых нечитающих дуг, удаляемых алгоритмом 1. В новом автомате P остается начальной вершиной единственной нечитающей дуги. Таким образом, операция $Automaton$ не вносит недетерминированности.

2. Из доказательства леммы 2 имеем, что если предложение ДМА содержит нечитающий нейтральный или накапливающий цикл, то любая дуга следующего за этим циклом участка совпадает с некоторой его дугой. Таким образом, предложение автомата M имеет вид TT', где T не содержит дуг циклов из $C_+(M)$, а T' либо пуст, либо каждая его дуга есть дуга некоторого цикла из $C_+(M)$. В автомате M' маршруту TT' отвечает в первом случае маршрут $T = TT'$, а во

втором случае маршрут $T(p,Z)\Lambda(f,Z)$, если $end(T)=(p,Z)$ и $p\notin F$, или просто T, если $p\in F$. Отсюда следует равенство языков $L(M)$ и $L(M')$.

3. Из конструкции автомата M' получаем равенство $C_+(M')=\emptyset$. По лемме 2 M' не имеет зацикливающих конфигураций □

Пример 1. Магазинный автомат с заключительным состоянием p и следующими командами допускает язык $\{a\}$:

1) $(p_0,a,Z_0)\to(p_0,Z_0a)$,
2) $(p_0,\Lambda,a)\to(p,aZ)$,
3) $(p,\Lambda,Z)\to(p_0,\Lambda)$.

Соответствующий автомату граф имеет три дуги:

$$(p_0,Z_0)\frac{a}{+a}(p_0,a),\ (p_0,a)\frac{\quad}{+Z}(p,Z),\ (p,Z)\frac{\quad}{-Z}(p_0,a).$$

После устранения нечитающих концевых циклов остается только первая из них. Ее конечная вершина становится заключительной, но состояние, входящее в состав этой вершины, не является заключительным состоянием исходного автомата. Состояние p_0 нельзя объявить заключительным: результирующий граф будет определять язык $\{\Lambda,a\}$ вместо исходного. Операция $Automaton$ добавляет новое состояние f и новую команду

$$(p_0,\Lambda,a)\to(f,a)$$

и тем решает проблему.

1.3. Автоматы, наполняющие магазин в реальное время

Пусть магазинный автомат M таков, что $\omega(\pi)\neq\Lambda$ для любой дуги $\pi\in E(M)$. Тогда назовем M *магазинным автоматом реального времени* (МАРВ). Будем также говорить, что M *действует в реальное время*.

Утверждение 1. Языки, допускаемые детерминированными МАРВ, составляют собственный подкласс детерминированных языков □

Для доказательства достаточно указать пример детерминированного языка, не допускаемого никаким детерминированным МАРВ (ДМАРВ). Таков язык

$$\{a^mb^na^m\mid m,n\geq 0\}\cup\{a^mb^ncb^na^m\mid m,n\geq 0\}$$

(см. [Ginsburg -Greibach 66]).

Утверждение 1 означает, что детерминированных МАРВ недостаточно для описания класса детерминированных языков. Остается строить автоматы, по возможности близкие к форме ДМАРВ.

Пусть граф

$$\mathcal{G}=(K,\Sigma,\Gamma,Z_0,V,E,p_0,Fin)$$

таков, что для любой дуги $\pi\in E$ из равенства $\omega(\pi)=\Lambda$ следует равенство $\mu(\pi)=-Z$ для некоторого Z из Γ. Тогда назовем автомат $Automaton(\mathcal{G})$ *наполняющим магазин в реальное время*.

Заметим, что ДАНРВ может иметь нечитающие нестирающие дуги. Но каждое предложение ДАНРВ содержит не более одной нечитающей нестирающей дуги. Такая дуга, если она есть, заканчивает предложение, как в случае ДАНРВ $M=$

$Automaton(\mathcal{G})$ для

$$\mathcal{G} = (\{p\}, \{a, b\}, \Gamma, Z_0, V, E, p, \{(p, X)\}),$$

где $\Gamma = \{X, Y, Z,\ Z_0\}$, $V = \{(p, W) \mid W \in \Gamma\}$, множество E составлено дугами

$$(p, Z_0)\frac{a}{+XZ}(p, Z),\ (p, Z_0)\frac{b}{+YZ}(p, Z),\ (p, Z)\frac{a}{-Z}(p, X),\ (p, Z)\frac{a}{-Z}(p, Y).$$

Граф $ExtG(M)$ имеет дополнительно к дугам графа $\mathcal{G}$ дугу

$$(p, X)\Lambda(f, X).$$

Докажем, что детерминированный язык допускается некоторым ДАНРВ.

Введем обозначение

$$E_+(M) = \{\pi \in E(M) \mid \omega(\pi) = \Lambda, \exists\gamma \in \Gamma^*\ \mu(\pi) = +\gamma\}.$$

Если G — граф, эквивалентный M, то наряду с обозначением $E_+(M)$ будем использовать обозначение $E_+(G)$.

Пусть $\pi \in E(M)$. Тогда назовем множество маршрутов

$$adj(\pi) = \begin{cases} \{end(\pi)\},\ \nexists\pi' \in E(M) : end(\pi) = beg(\pi'), \\ \{\pi' \in E(M) | end(\pi) = beg(\pi')\} \text{ в противном случае} \end{cases}$$

множеством *соседей дуги* π.

Заметим, что если $\pi' \in adj(\pi)$, то $\pi\pi'$ не обязательно является маршрутом. Так, в графе, содержащем дуги

$$\pi = (p, Z)\frac{b}{-Z}(q, Y),$$

$$\pi_1 = (q_1, X)\frac{\Lambda}{+Z}(p, Z),$$

$$(p, Z)\frac{a}{-Z}(q, X),$$

$$(q_2, Y)\frac{\Lambda}{+Z}(p, Z),$$

дуга π принадлежит множеству $adj(\pi_1)$, но $\pi_1\pi$ не является маршрутом.

Пусть $\epsilon \in E_+(M)$, $T \in adj(\epsilon)$. Тогда назовем *вычеркиванием* функцию $\varepsilon(\epsilon, T)$, определяемую следующим образом: если T — пустой маршрут, то

$$\varepsilon(\epsilon, T) = beg(\epsilon),$$

иначе

$$\varepsilon(\epsilon, T) = beg(\epsilon)\frac{\omega(T)}{\mu(\alpha(\epsilon)\mu(T))}end(T).$$

Алгоритм 2. Устранение нечитающей нестирающей дуги.

Вход. Магазинный автомат

$$M = (K, \Sigma, \Gamma, Z_0, p_0, F)$$

и дуга

$$\epsilon \in E_+(M).$$

Выход. Граф $Eps(ExtG(M), \epsilon) = (K,\ \Sigma,\ \Gamma,\ Z_0,\ V,\ E,\ p_0,\ Fin)$.

Метод. $E := E(M) - \{\epsilon\} \cup \{\varepsilon(\epsilon, \pi) | \pi \in adj(\epsilon) - \{end(\epsilon\}\}$;
$V := \{P_0(M)\} \cup \{P \in V(M) \mid \exists \pi \in E\ P$ инцидентна дуге $\pi\}$;
$Fin := V \cap [Fin(M) \cup \{beg(\epsilon) \mid end(\epsilon) \in Fin(M)\}]$.

Начальный участок предложения назовем *инициальным маршрутом*. Используем этот термин в следующей лемме, которая доказывает эквивалентность графа, создаваемого алгоритмом 2, входному автомату.

Лемма 3. Пусть M — магазинный автомат, $\epsilon \in E_+(M)$, $L = L(Eps(ExtG(M),\ \epsilon))$. Тогда $L = L(ExtG(M))$. Следовательно, $L = L(M)$.

Доказательство. Достаточно указать, что существует сюръективное отображение множества инициальных маршрутов графа $G = ExtG(M)$ на множество инициальных маршрутов графа $G' = Eps(G,\ \epsilon)$, при котором маршрут T графа G и его образ T' имеют одинаковые заряды и пометки, а вершины $end(T)$ и $end(T')$ одновременно являются или не являются заключительными.

Произвольный маршрут T в графе G можно представить в виде

$$T = T_0\epsilon\pi_1T_1 \ldots \epsilon\pi_nT_n,$$

где $n \geq 0$, T_0 не содержит дуги ϵ, для $i = 1, \ldots, n$ участок π_iT_i не содержит дуги ϵ и либо пуст, либо $\pi_i \in adj(\epsilon)$. Расширим понятие вычеркивания следующим образом: $\varepsilon(\epsilon,\ T) = T_0\varepsilon(\epsilon,\ \pi_1)T_1 \ldots \varepsilon(\epsilon,\ \pi_n)T_n$. Оно и будет искомым сюръективным отображением. В самом деле, из конструкции графа G' следует, что $T' = \varepsilon(\epsilon, T)$, где T — инициальный маршрут графа G, есть инициальный маршрут в G', причем $\omega(T') = \omega(T)$, $\mu(T') = \mu(T)$ и вершины $end(T)$ и $end(T')$ одновременно являются или не являются заключительными. инициальных маршрутов, не принадлежащих множеству

$$\{\varepsilon(\epsilon, T) \mid T \text{ — инициальный маршрут графа } G\},$$

граф G', по построению, не имеет □

Лемма 4. Если M есть ДМА и $\epsilon \in E_+(M)$, то

$$Automaton(Eps(ExtG(M),\ \epsilon))$$

является ДМА.

Доказательство. Дуга ϵ не является стирающей, так как она принадлежит множеству $E_+(M)$. Вследствие детерминированности автомата M вершина $beg(\epsilon)$ является начальной вершиной только одной дуги — нечитающей нестирающей дуги ϵ. Алгоритм 2 заменяет эту дугу некоторым подмножеством $A \subseteq \{\varepsilon(\epsilon,\ \epsilon\pi) | \pi \in adj(\epsilon)\}$. Дуги множества A отличаются от дуг автомата M, имеющих начальную вершину $end(\epsilon)$, только своей начальной вершиной. Предположение, что изза дуг множества A нарушается свойство детерминированности, означало бы, что оно нарушается и дугами из $adj(\epsilon)$. Однако исходный автомат является детерминированным □

Из теоремы 1 следует, что можно без ограничения общности предполагать однозначность рассматриваемых ДМА.

Алгоритм 3. Преобразование ДМА в ДАНРВ.

Вход. Однозначный ДМА $M = (K, \Sigma, \Gamma, Z_0, \delta, p_0, F)$.

Выход. Магазинный автомат

$$RealTimeCharging(M) = (K', \Sigma, \Gamma, Z_0, \delta', p_0, F').$$

Шаг 1. $G := ExtG(M)$.

Шаг 2. Пока $E_+(G) \neq \emptyset$, выполнять следующие два действия:
выбрать дугу $\epsilon \in E_+(G) \cap \{\pi \mid adj(\pi) \cap E_+(G) = \emptyset\}$;
$G := Eps(G, \epsilon)$.

Шаг 3. Положить автомат $RealTimeCharging(M)$ равным $Automaton(G)$.

Лемма 5. Пусть M — однозначный ДМА и $E_+(M) \neq \emptyset$. Тогда существует такая дуга $\epsilon \in E_+(M)$, что $adj(\epsilon) \cap E_+(M) = \emptyset$.

Доказательство. Предположим, что для любой $\epsilon \in E_+(M)$ пересечение $adj(\epsilon) \cap E_+(M)$ непусто. Тогда существует последовательность ϵ_1, ..., ϵ_{n+1}, где $n = |E_+(M)|$, $\epsilon_1 \in E_+(M)$ и $\epsilon_{j+1} \in adj(\epsilon_j) \cap E_+(M)$ для $j = 1, \ldots, n$. Отсюда имеем соотношения $end(\epsilon_j) = beg(\epsilon_{j+1})$, $j = 1, \ldots, n$. Так как, кроме того, каждая дуга множества $E_+(M)$ является нейтральной или накапливающей, $\epsilon_1 \ldots \epsilon_{n+1}$ есть маршрут. Так как его длина больше $|E_+(M)|$, для некоторых k и l, $1 \leq k < l \leq n+1$, имеет место равенство $\epsilon_k = \epsilon_l$. Но тогда $\epsilon_k \ldots \epsilon_{l-1}$ есть нечитающий нейтральный или накапливающий цикл, что противоречит однозначности автомата M □

Теорема 2. Если M — однозначный ДМА, то:
1) алгоритм 3 заканчивает работу;
2) $RealTimeCharging(M)$ является ДМА и эквивалентен M.

Доказательство. Первое утверждение теоремы следует из леммы 5, второе — из лемм 3 и 4 □

§2. Ядро бесконтекстной грамматики и синтаксический анализ

Слово "грамматика"в данном параграфе обозначает приведенную [Aho -Ullman 72] бесконтекстную грамматику

$$G = (N,\ \Sigma,\ P,\ S),$$

где N — нетерминальный алфавит, Σ — терминальный алфавит, $S \in N$ — начальный символ, P — правила. Положим $V = N \cup \Sigma$. Условимся обозначать нетерминальные символы большими, а терминальные маленькими буквами начала латинского алфавита.

Здесь рассматриваются только правовыводимые сентенциальные формы. Для краткости они упоминаются как сентенциальные формы. Алгоритм синтаксического анализа обозначается словом "анализатор".

Пункт 2.1 вводит понятия ядра и характеристики бесконтекстной грамматики. Характеристика является модификацией ядра, удобной для построения недетерминированного анализатора, работающего как LR(1)анализатор для каждого из случаев разбора анализируемой цепочки. В пункте 2.2 определяется сам анализатор и доказывается, что он корректен. В пункте 2.3 рассматриваются приемы оптимизации анализатора, обоснование которых базируется на представлении

анализатора преобразователем с магазинной памятью и, следовательно, определенным образом обобщенным, "преобразующим"Dграфом. Здесь затрагивается один из сложнейших и интереснейших вопросов теории синтаксического анализа — вопрос уменьшения таблиц, используемых LR(k)анализатором. Наш подход вскрывает суть явлений, обусловливающих громоздкость LR(k)таблиц, и указывает границы достижимого при сокращении этих таблиц. Отмечается, что вместе с приемом расщепления грамматики, разработанным под руководством автора, сформулированные в пункте 2.3 приемы дают практически приемлемый метод построения LR(1)анализаторов. При этом наш метод применим к любой LR(1)-грамматике, а его эффект заключает в себе действие ранее известных способов оптимизации LR(1)анализаторов (например, оптимизация анализатора, предусматриваемая LALR(1)методом, полностью учитывается нашим методом).

2.1. Ядро и характеристика бесконтекстной грамматики. Для бесконтекстной грамматики определяется понятие структуры. Это один из вариантов линйной записи дерева вывода. Таким образом, структура представляет информацию, которую извлекает анализатор, причем представляет удачно в том смысле, что весьма точно указывает схему вычисления, выполняемого анализатором-преобразователем с магазинной памятью.

В терминах структур для грамматик определяются понятия канона и ядра, аналогия которых с одноименными понятиями, введенными в главе 1, достаточно ясна. В целом получается система понятий, позволяющая доказать известный факт об эквивалентности бесконтекстных грамматик и магазинных автоматов такими построениями, которые обеспечивают тесное соответствие всех особенностей эквивалентных устройств и, главное, возможность дотошно выяснить это соответствие. В данном параграфе фактически дается построение по бесконтекстной грамматике эквивалентного магазинного автомата. Ему в определенном смысле родственно построение грамматики по автомату, рассмотренное в [Вылиток 98].

2.1.1. Структуры. Каноны

Назовем грамматику

$$Augm(G) = (N \cup \{S_0\}, \Sigma \cup \{\perp\}, P \cup \{S_0 \to \perp S \perp\}, S_0),$$

где $S_0, \perp \notin V$, *пополнением* грамматики

$$G = (N, \Sigma, P, S).$$

Правилу

$$S_0 \to \perp S \perp$$

грамматики $Augm(G)$ сопоставим номер 0. Пусть $\mid P \mid = s$. Занумеруем правила грамматики G в некотором порядке номерами $1, \ldots, s$. Обозначим pе правило грамматики $Augm(G)$ записью

$$A_p \to X_{p1} \ldots X_{pn_p}.$$

Здесь

$$0 \le p \le s,\ n_p \ge 0,\ X_{pj} \in V \cup \{S_0, \perp\} \text{ для } 1 \le j \le n_p.$$

Обозначим через

$$Struct(G)$$

грамматику, определяемую правилами

$$\{A_p \to \langle_p X_{p1} \dots X_{p_{n_p}} \rangle_p \mid 0 \le p \le s;\ \langle_p, \rangle_p \text{ — новые символы}\}.$$

О выводе

$$x_1 \Rightarrow_G \dots \Rightarrow_G x_m,\ m \ge 1,$$

будем говорить, что он *реализует соотношение*

$$x_1 \Rightarrow^*_G x_m.$$

Напомним, что правый вывод, реализующий соотношение $A \Rightarrow^*_G x$, может быть представлен последовательностью правил или их номеров.

Ясно, что между правилами грамматик $Augm(G)$ и $Struct(G)$ существует взаимнооднозначное соответствие. Будем считать, что соответствующие правила имеют в своих грамматиках один и тот же номер. Используем это обстоятельство в следующем определении.

Пусть $A \in N \cup \{S_0\}$, пусть соотношения

$$A \Rightarrow^*_{Augm(G)} x$$

и

$$A \Rightarrow^*_{Struct(G)} \sigma$$

реализуются одним и тем же выводом (последовательностью номеров правил) П. Тогда будем говорить, что σ является *структурой* цепочки x.

Если σ — структура цепочки x, то будем писать $\sigma \in struct(x)$, $x = \omega(\sigma)$.

Обозначим через $br(\sigma)$ (от braces) результат вычеркивания из σ символов цепочки $\omega(\sigma)$. Заметим, что $br(\sigma)$ является скобочной системой над D-множеством

$$\{(\langle_p, \rangle_p) \mid 0 \le p \le s\}.$$

Будем применять операцию br и к подцепочкам структур.

О подцепочке z некоторой структуры будем говорить, что она *сбалансирована*, если $br(z)$ — скобочная система.

Пусть $1 \le p \le s$; пусть

$$\sigma = u\langle_p x\langle_p z\rangle_p y\rangle_p v$$

есть структура, в которой z и $w = \langle_p x\langle_p z\rangle_p y\rangle_p$ сбалансированы. Тогда будем говорить, что цепочка w *определяет p-итерацию (x, z, y) с циклами x и y*. При этом p-итерацию будем называть *элементарной*, если для любого $1 \le q \le s$ никакая подцепочка цепочки $x\langle_p z\rangle_p y$ не определяет q-итераций.

Термин p-итерация отражает повторение в выводе применений p-го правила. Приставку p будем опускать, когда правило несущественно.

Пусть для $1 \le i \le 5$ подцепочка

$$\langle_p z_i\rangle_p = \langle_p x_i \dots \langle_p x_5\langle_p z_6\rangle_p y_5\rangle_p \dots y_i\rangle_p$$

цепочки

$$w = \langle_p x_1 \dots \langle_p x_5\langle_p z_6\rangle_p y_5\rangle_p \dots y_1\rangle_p$$

определяет p-итерацию (x_i, z_{i+1}, y_i). Тогда назовем любую из p-итераций, определяемых цепочкой w, *сложной*. Итерацию, которая не является сложной, назовем *простой*.

Пусть каждая итерация, определяемая подцепочками структуры σ, является простой. Тогда назовем σ *каноном*.

Множество сентенциальных форм грамматики $Struct(G)$ обозначим через $\mathcal{S}_G$. Введем обозначение $Canons(G) = \{\sigma \in \mathcal{S}_G \mid \sigma \text{ является каноном}\}$.

2.1.2. Конечность множества канонов. Докажем одно утверждение о специальном подмножестве Dязыка $\mathcal{L}_P$, полезное для исследования свойств канонов грамматики.

Пусть l — положительное целое, $selection(\mathcal{L}_P, l) = \{\psi \in \mathcal{L}_P | \forall (a,b) \in \mathcal{P}\ max\{n|\ \exists\{x_i,\ y_i \notin \mathcal{L}_P | 1 \le i \le n, ax_iy_ib, ax_i \dots x_ny_n \dots y_ib \in \mathcal{L}_P\} : \psi = ax_1 \dots ax_ny_nb \dots y_1b\} \le l\}$.

Утверждение 2. Если $\psi \in selection(\mathcal{L}_P, l)$, то $depth(\psi) \le l|\mathcal{P}|$.

Доказательство. Пусть $n = depth(\psi)$, $m = |\mathcal{P}|$. Не ограничивая общности, можно полагать, что ψ имеет вид $a_1 \dots a_nb_n \dots b_1$, где $(a_i, b_i) \in \mathcal{P}$ для $1 \le i \le n$.

Докажем утверждение индукцией по m. В случае $m = 1$ из определения множества $selection(\mathcal{L}_P, l)$ следует, что $n \le l = lm$. Пусть $k > 1$. Предположим, что утверждение верно для $1 \le m < k$, и рассмотрим случай $m = k$. Обозначим через ξ цепочку Dязыка над Dмножеством $\mathcal{P} - \{(a_1, b_1)\}$, получаемую из ψ вычеркиванием всех вхождений парных символов a_1, b_1. Согласно определению множества $selection(\mathcal{L}_P, l)$ имеем $depth(\xi) \ge n - l$. По предположению индукции $depth(\xi) \le l(m-1)$. Следовательно, $n - l \le l(m-1)$, т. е. $n \le lm$ □

Заметим, что в случае языка Дика мощность Dмножества совпадает с числом открывающих скобок (или, что то же, с числом закрывающих скобок).

Пусть r — максимальная из длин правых частей правил грамматики G, т.е.

$$r = max\{n_p \mid 0 \le p \le s\}.$$

Из построения грамматики $Struct(G)$ следуют

Лемма 6. Подцепочка $y \in (V \cup \{\perp\})^*$ структуры удовлетворяет неравенству $|\ y\ | \le r$ □

Лемма 7. Для $z \in V^*$ соотношение $x\langle_p z\rangle_p y \in \mathcal{S}_G$ верно тогда и только тогда, когда $A_p \to z \in P$ и $xA_py \in \mathcal{S}_G$ □

Глубину и ширину скобочной системы $br(\sigma)$ будем кратко называть *глубиной и шириной структуры* σ.

Лемма 8. Глубина канона не превосходит $5(s+1)$.

Доказательство. Справедливо соотношение

$$\{br(\sigma)|\sigma \in Canons(G)\} \subseteq selection(\mathcal{L}_{\{\langle_p,\rangle_p|0\le p\le s\}}, 5),$$

так как канон не содержит сложных итераций. Но тогда лемма следует из утверждения 2 □

Лемма 9. Ширина канона не превосходит r.

Доказательство. Пусть структура σ является каноном. Предположим, что ее ширина w больше r. Тогда σ представляется в виде

$$\sigma = u\langle_{p_0} x_1 \langle_{p_1} y_1 \rangle_{p_1} \dots x_w \langle_{p_w} y_w \rangle_{p_w} x_0 \rangle_{p_0} v,$$

где $br(x_i) = \Lambda$ (т. е. x_i не содержит угловых скобок) для $0 \le i \le w$ и цепочка y_j сбалансирована для $1 \le j \le w$. Но, по лемме 7, цепочка

$$u\langle_{p_0} x_1 A_{p_1} \dots x_w A_{p_w} x_0 \rangle_{p_0} v$$

также является структурой, и

$$x_1 A_{p_1} \dots x_w A_{p_w} x_0$$

есть правая часть правила p_0. Но такое правило противоречит определению числа r, так как

$$| x_1 A_{p_1} \dots x_w A_{p_w} x_0 | \ge w > r$$

□

Теорема 3. Если $\sigma \in Canons(G)$, то

$$| \sigma | \le (r+1) g_{r,5(s+1)} - r.$$

Следовательно, множество $Canons(G)$ конечно.

Доказательство. Из лемм 8, 9 и теоремы 1.1 об оценке сверху длины скобочной системы с ограниченными глубиной и шириной следует, что число угловых скобок в структуреканоне не превосходит числа $g_{r,5(s+1)}$. Так как первый и последний символы в структуре из $Canons(G)$ являются скобками $\langle_0$ и $\rangle_0$ соответственно, она содержит не более $g_{r,5(s+1)} - 1$ подцепочек вида $b_1 x b_2$, где b_1 и b_2 — скобки, а x не содержит скобок. Теперь из леммы 6 следует, что структураканон содержит не больше $r(g_{r,5(s+1)} - 1)$ символов из $V \cup \{\perp\}$. Откуда и следует теорема □

2.1.3. Определение ядра. Теорема о развитии для бесконтекстных грамматик. Далее удобно называть структурой и любую часть цепочки из $\mathcal{S}_G$.

Следующая функция отображает структуру во множество структур. Если σ не имеет итераций, то

$$DelPair(\sigma) = \{\sigma\},$$

иначе

$$DelPair(\sigma) = \{u\langle_p z\rangle_p v \mid \exists (x,y) : \sigma = u\langle_p x\langle_p z\rangle_p y\rangle_p v$$
$$\text{и } \langle_p x\langle_p z\rangle_p y\rangle_p \text{ определяет p-итерацию } (x,z,y)\}.$$

Можно сказать, что операция $DelPair$ вычеркивает циклы некоторой итерации. Так как структура может иметь не одну итерацию, такими вычеркиваниями можно получить множество структур, содержащее больше одного элемента.

Следующая функция отображает разбиение структуры во множество структур. Пусть xyz — структура; тогда если циклы некоторой итерации "расположены"один в x, другой в z, то

$$PrDel(x,y,z) = \{x_1\langle_p x_2 y z_2\rangle_p z_1 \mid 1 \le p \le s,$$
$$\exists (u,v) : \ (x = x_1\langle_p u\langle_p x_2,\ z = z_2\rangle_p v\rangle_p z_1,$$
$$(u, x_2 y z_2, v) \text{ — p-итерация })\},$$

иначе

$$PrDel(x,y,z) = \{xyz\}.$$

$PrDel$ отличается от $DelPair$ тем, что "не затрагивает"вычеркиваниями выделенный участок y заданной структуры xyz.

Пусть $m \geq 1$, $\sigma = y_1 x_1 \dots y_{m-1} x_{m-1} y_m$ — структура. Пусть $Step = \{u_j z_j v_j \mid 1 \leq j \leq m,\ u_j y_j v_j = \sigma,\ z_j \in DelPair(y_j)\} \cup \{u_i z_i w_{ij} z_j v_j \mid 1 \leq i < j \leq m,\ u_i y_i w_{ij} y_j v_j = \sigma,\ z_i w_{ij} z_j \in PrDel(y_i, w_{ij}, y_j)\}$. Тогда если $Step = \{\sigma\}$, то $reduction(y_1, x_1, \dots, y_{m-1}, x_{m-1}, y_m) = \{\sigma\}$, иначе $reduction(y_1, x_1, \dots, y_{m-1}, x_{m-1}, y_m) = \cup_{z_1 x_1 \dots z_{m-1} x_{m-1} z_m \in Step} reduction(z_1, x_1, \dots, z_{m-1}, x_{m-1}, z_m)$.

"Редукция"приводит структуру σ к виду

$$u_1 x_1 \dots u_{m-1} x_{m-1} u_m,$$

где никакая из цепочек u_i не содержит итераций и никакая пара (u_i, u_j) не содержит циклов одной и той же итерации.

Из определения функции $reduction$ видно, что она "укорачивает"вывод своего аргумента, уменьшая в выводе число повторных применений правил. Таким образом, справедлива

Лемма 10. Если

$$y_1 x_1 \dots y_{m-1} x_{m-1} y_m \in \mathcal{S}_G,$$

то для любой

$$\sigma \in reduction(y_1, x_1, \dots, y_{m-1}, x_{m-1}, y_m)$$

верно соотношение $\sigma \in \mathcal{S}_G$ $\square$

Определим на множестве структур следующие бинарные отношения.

$$(\sigma', \sigma) \in \uparrow_G \Longleftrightarrow \exists (p, x, y, z, u, v) : [\sigma' = u\langle_p z\rangle_p v;\ \sigma = u\langle_p x\langle_p z\rangle_p y\rangle_p v;$$

$\langle_p x\langle_p z\rangle_p y\rangle_p$ определяет элементарную p-итерацию $(x, z, y)]$.

Будем говорить, что σ *является развитием структуры* σ' *с историей (развития)*

$$(u, \langle_p x, \langle_p z\rangle_p, y\rangle_p, v).$$

Назовем отношение $\Uparrow_G = \uparrow_G^*$ *отношением развития*. Если последовательность $\sigma_0, \dots, \sigma_m$ такова,что для $1 \leq i \leq m$ σ_i является развитием структуры σ_{i-1} с некоторой историей $h_i = (u_i, x_i, z_i, y_i, v_i)$, то будем говорить, что последовательность $(h_1, \dots, h_m)$ *реализует соотношение* $\sigma_0 \Uparrow_G \sigma_m$.

Индекс G, как обычно, опускаем всегда, когда для этого есть возможность.

Лемма 11. Пусть $\sigma \in \mathcal{S}_G - Canons(G)$. Тогда некоторая подцепочка цепочки σ определяет элементарную итерацию.

Доказательство. Из определения канона следует, что σ содержит цепочку, определяющую итерацию. Выберем среди таких цепочек наименьшую по длине. Предположим, что выбранная цепочка w определяет неэлементарную p-итерацию. Тогда $w = \langle_p x\langle_p z\rangle_p y\rangle_p$ для некоторых x, y, z и некоторая подцепочка цепочки $x\langle_p z\rangle_p y$ определяет итерацию и имеет меньшую длину, чем w. Но это противоречит выбору цепочки w $\square$

Лемма 12. Пусть $\sigma \in \mathcal{S}_G$, $(\sigma_0, \sigma) \in \uparrow$ с историей $(u, \langle_p x, \langle_p z\rangle_p, y\rangle_p, v)$. Тогда существуют такие структуры u', v', что $u'\langle_p x\langle_p z\rangle_p y\rangle_p v' \in Canons(G)$.

Доказательство. Рассмотрим структуру

$$\sigma' = u'\langle_p x\langle_p z\rangle_p y\rangle_p v' \in reduction(u, \langle_p x\langle_p z\rangle_p y\rangle_p, v).$$

Предположим, что σ' не является каноном. Тогда σ' содержит цепочку w_1 вида

$$w_1 = \langle_q x_1 \ldots \langle_q x_5 \langle_q z_1 \rangle_q y_5 \rangle_q \ldots y_1 \rangle_q,$$

определяющую сложную итерацию.

Согласно определениям элементарной итерации и развития цепочка $x\langle_p z\rangle_p y$ не может содержать в себе цепочки, определяющей итерацию. Следовательно, w_1 либо содержится целиком в u' или v', либо какая-либо из цепочек $\langle_q z_1\rangle_q$, $x_5\langle_q z_1\rangle_q y_5$, x_i, y_i, $i = 1, 2, 3, 4, 5$, содержит в себе цепочку $\langle_p x\langle_p z\rangle_p y\rangle_p$. В первом случае цепочка, содержащая в себе w_1, может быть уменьшена операцией $DelPair$. Во втором случае u' или v' (или они обе) могут быть уменьшены операцией $DelPair$ (или, соответственно, операцией $PrDel(u', \langle_p x\langle_p z\rangle_p y\rangle_p, v')$). Каждый из рассмотренных случаев противоречит "неуменьшаемости"цепочек u', v', следующей из определения операции $reduction$. Следовательно, σ' является каноном. Так как $\sigma \in \mathcal{S}_G$, из леммы 10 следует, что $\sigma' \in Canons(G)$ $\square$

Теорема 4. Для каждой структуры $\sigma \in \mathcal{S}_G$ существует такая $\sigma' \in Canons(G)$, что $(\sigma', \sigma) \in \Uparrow$ и для любого элемента (u, x, z, y, v) последовательности, реализующей последнее соотношение, xzy входит в некоторый канон из $Canons(G)$.

Доказательство. Достаточно рассмотреть структуру σ, имеющую сложную итерацию. По лемме 11 σ содержит цепочку, определяющую элементарную итерацию. Пусть

$$\sigma = u\langle_p x\langle_p z\rangle_p y\rangle_p v,$$

где $\langle_p x\langle_p z\rangle_p y\rangle_p$ определяет элементарную p-итерацию (x, z, y). Пусть

$$\sigma_0 = \sigma,\ \sigma_1 = u\langle_p z\rangle_p v.$$

Тогда $(\sigma_1, \sigma_0) \in \uparrow$ с историей

$$h_1 = (u_1, x_1, z_1, y_1, v_1) = (u, \langle_p x, \langle_p z\rangle_p, y\rangle_p, v).$$

По лемме 12 $x_1 z_1 y_1$ входит в некоторый канон из $Canons(G)$.

Если σ_1 имеет сложную итерацию, то аналогичным образом строим такую структуру σ_2, что

$$(\sigma_2, \sigma_1) \in \uparrow \text{ с историей } h_2 = (u_2, x_2, z_2, y_2, v_2),$$

в которой $x_2 z_2 y_2$ — подцепочка некоторого канона из $Canons(G)$, и т.д. Так как $|\ \sigma_i\ | < |\ \sigma_{i-1}\ |$ для $i \geq 1$, процесс закончится для некоторого $k \geq 1$ построением структуры σ_k, не имеющей итераций, и такой, что $(\sigma_k, \sigma) \in \uparrow^k$. При этом в последовательности $(h_k, \ldots, h_1)$, реализующей соотношение $(\sigma_k, \sigma) \in \Uparrow$, для каждого $h_i = (u_i, x_i, z_i, y_i, v_i)$, $1 \leq i \leq k$, $x_i z_i y_i$ входит в некоторый канон из $Canons(G)$. Вследствие леммы 10 $\sigma_i \in \mathcal{S}_G$, $1 \leq i \leq k$. Следовательно, $\sigma_k \in Canons(G)$ $\square$

Пусть структура σ такова, что $\omega(\sigma) \in (\Sigma \cup \{\perp\})^*$. Тогда назовем *$\sigma$ терминальной структурой.*

Назовем *ядром грамматики* G множество

$$Core(G) = \{\sigma \in Canons(G) \mid \sigma \text{ — терминальная структура}\}.$$

Из доказательств теоремы 4 и леммы 12 следует

Теорема 5. Для каждой терминальной структуры $\sigma \in \mathcal{S}_G$ существует такая $\sigma' \in Core(G)$, что $(\sigma', \sigma) \in \Uparrow$ и для любого элемента (u, x, z, y, v) последовательности,

реализующей последнее соотношение, xzy входит в некоторый канон из $Core(G)$
□

Заметим, что во многих случаях было бы вполне удобно такое определение простой итерации, в котором вместо числа пять фигурирует число два. С другой стороны, число пять можно было бы заменить любым большим. Наш выбор границы, разделяющей итерации на простые и сложные, оправдан следующим.

По нашему замыслу, каждая терминальная цепочка, определяющая простую итерацию, обязана присутствовать в некотором элементе ядра, а ядро должно исчерпывающим образом характеризовать порождающую способность любого нетерминального символа грамматики. В частности, ядро должно указывать, является ли нетерминальный символ самовставляющимся.

Самовставляющийся символ неизбежно порождает цепочки, определяющие итерации. Естественно по таким именно цепочкам проверять, имеет ли место самовставление (другими словами, участвует ли операция согласованной итерации в бесконтекстном выражении, связанном с данным ядром). Учтем также, что если A_p — самовставляющийся символ, порождающий цепочку xA_py, где x и y — непустые терминальные цепочки, то появление символов цепочек x и y может быть обусловлено разными правилами грамматики. Изза этого явление самовставления может обнаруживаться не по любой p-итерации, а лишь по достаточно сложно устроенным. В самом деле, рассмотрим следующее утверждение и устройство обсуждаемых в нем терминальных структур.

Утверждение 3. Пусть $\sigma = uwv$ есть терминальная структура, в которой цепочка w определяет p-итерацию (x_1ax_2, z, y_1by_2), где $a, b \in \Sigma$. Тогда существует терминальная структура $\sigma' = u'w'v' \in Core(G)$, в которой цепочка w' определяет p-итерацию $(x'_1ax'_2, z', y'_1by'_2)$.

Доказательство. Пусть

$$\sigma' \in reduction(u, \langle_p, x_1, a, x_2, \langle_p, z, \rangle_p, y_1, b, y_2, \rangle_p, v).$$

Тогда σ' представляется в виде $\sigma' = u'w'v'$, где

$$w' = \langle_p x'_1 a x'_2 \langle_p z' \rangle_p y'_1 b y'_2 \rangle_p$$

определяет p-итерацию требуемого вида.

Предположим, что σ' не является каноном. Это означает, что некоторая подцепочка

$$s' = \langle_q s_5 \ldots \langle_q s_1 \langle_q r \rangle_q t_1 \rangle_q \ldots t_5 \rangle_q$$

структуры σ' определяет сложную q-итерацию. Но при любом положении в σ' скобок $\langle_q, \rangle_q$ цепочки s' получается, что в цепочке σ', вопреки ее построению, еще можно удалить циклы какой-либо из пар (s_i, t_i), $1 \le i \le 5$, не затрагивая шести выделенных участков $\langle_p$, a, $\langle_p$, $\rangle_p$, b, $\rangle_p$. Итак, σ' является каноном и, вследствие леммы 10, входит в $Core(G)$ □

Заметим, что цепочка вида

$$s' = \langle_q s_4 \ldots \langle_q s_1 \langle_q r \rangle_q t_1 \rangle_q \ldots t_4 \rangle_q,$$

определяющая простую итерацию, может присутствовать в σ', если s_4 содержит первую, а s_1 — вторую из выделенных открывающих скобок $\langle_p$, t_4 и t_1 содержат

соответствующие закрывающие скобки и, например, s_3 содержит выделенное вхождение символа a, а t_2 — выделенное вхождение символа b (при этом x'_2 содержит s_2 как подцепочку, y'_2 содержит t_3).

2.1.4. Прогнозы. Характеристика бесконтекстной грамматики

Пусть $0 \le p \le s$. Тогда назовем пару (p, j), $1 \le j \le n_p$, *позицией* (в правилах грамматики $Augm(G)$) символа X_{pj}. Если $n_p = 0$, то определим $(p, 0)$ как позицию пустой цепочки.

Заметим, что одному символу грамматики может отвечать несколько позиций. То же верно для пустой цепочки. Для $X \in V \cup \{\bot, \Lambda\}$ обозначим через $positions(X)$ множество его позиций. Будем писать $symb(r) = X \ \forall r \in positions(X)$.

Определим рекурсивно *роли* некоторых подцепочек структуры $\sigma \in \mathcal{S}_G$ (в этой структуре):

1) пусть

$$\sigma = u\langle_p v\rangle_p \beta a z \in L(Struct(G)),$$

$0 \le p \le s$, цепочка v сбалансирована, $a \in \Sigma \cup \{\bot\}$ или $az = \Lambda$, $\omega(\beta) = \Lambda$; тогда если $v = \Lambda$, то пустая цепочка между скобками $\langle_p$ и $\rangle_p$ имеет в σ роль $[p, 0, a]$; если же $\Lambda \ne v = x_1 \dots x_{n_p}$, где $X_{pj} \Rightarrow^*_{Struct(G)} x_j$ для $1 \le j \le n_p$, то цепочка x_j имеет в σ роль $[p, j, a]$;

2) пусть $u\langle_p v\rangle_p y \in \mathcal{S}_G$ и выделенная подцепочка $\langle_p v\rangle_p$ имеет в данной структуре роль r; тогда символ A_p имеет роль r в структуре $\sigma = uA_p y$.

Как видим, в состав роли входят позиция в некотором правиле и терминальный символ, непосредственно следующий в некоторой сентенциальной форме грамматики $Augm(G)$ за символом левой части этого правила.

Рассмотрим вопрос построения множества $roles(X)$ всех ролей элемента $X \in V \cup \{\bot, \Lambda\}$. Нетрудно убедиться, что структура

$$reduction(u, \langle_p, v, \rangle_p, \beta, a, z)$$

является каноном. Следовательно, для такого построения достаточно ядра $Core(G)$.

Подмножество множества

$$\mathcal{R} = \cup_{X \in V \cup \{\bot, \Lambda\}} roles(X)$$

назовем *прогнозом*.

Понятия роли и прогноза будут основными в нашем исследовании вопросов синтаксического анализа. Это естественно, так как задача анализатора — определить, явно или неявно, роль каждого символа рассматриваемых им сентенциальных форм. Однако, вообще говоря, анализатор располагает только прогнозом.

Термин "прогноз"призван заменить громоздкие словосочетания, введенные в работах по теории LR(k)анализа. Термин не только краток, но и достаточно метко характеризует информацию, хранимую LR(k)анализатором в магазине.

Обозначим через $\rho(\sigma)$, где $\sigma \in \mathcal{S}_G$, результат вставления в σ вслед за каждой ее подцепочкой, имеющей в σ роль, этой роли.

Пусть $m \ge 1$, $\sigma = u_1 \dots u_m \in \mathcal{S}_G$. Пусть для $1 \le i \le m$ имет место равенство $u_i = u_{i1} \dots u_{ik_i}$, где для $1 \le j \le k_i$ $u_{ij} \in V \cup \{\bot\} \cup \{\langle_p, \rangle_p | 0 \le p \le s\}$. Тогда обозначим через $\rho(u_1, \dots, u_m)$ такую последовательность $(u'_1, \dots, u'_m)$, что $\rho(\sigma) = u'_1 \dots u'_m$,

$u'_i = u_{i1}r_{i1}\dots u_{ik_i}r_{ik_i}$, где $r_{ij} \in \mathcal{R} \cup \{\Lambda\}$. (Таким образом, цепочка u'_i содержит вместе с символами цепочки u_i их роли в структуре σ.) Заметим, что одна и та же подцепочка u_i в разных структурах или в разных местах одной структуры может отображаться операцией ρ на различные цепочки. Будем использовать для u'_i обозначение $\rho(u_i)$ в тех случаях, когда u'_i известна и, следовательно, исключена необходимость выбирать среди цепочек, могущих иметь обозначение $\rho(u_i)$.

Заметим, что цепочка ролей, получаемая из $\rho(\sigma)$ вычеркиванием всех символов исходной структуры σ, позволяет восстановить σ. Таким образом, в $\rho(\sigma)$ символы исходной структуры несут лишь избыточную информацию, и потому мы будем иногда игнорировать их (см., например, определение 22). Однако, они удобны, например, для определения прогнозов, которые являются состояниями конструируемого ниже преобразователя $M(G)$.

Назовем множество

$$\chi(G) = \{\rho(\sigma) | \sigma \in Core(G)\}$$

характеристикой грамматики G.

Далее мы рассмотрим способы построения автомата, эквивалентного грамматике G, по характеристике $\chi(G)$. При некоторых способах построения автомат наследует особенности грамматики G, например, является однозначным, если G однозначна, и т.п.

2.2. ELR(1)анализатор бесконтекстного языка

2.2.1. Построение преобразователя по элементам характеристики, рассматриваемым порознь. Рассмотрим расширенный магазинный автомат

$$M = (K, \Sigma, \Gamma, Z_0, \delta, q_0, F),$$

который "видит"в магазине цепочку, а не один только верхний символ. Отображение δ интерпретируем как множество команд вида

$$(q_1, a, \gamma_1) \to (q_2, \gamma_2),$$

где $a \in \Sigma \cup \{\Lambda\}$ и для $i = 1, 2$ $q_i \in K$, $\gamma_i \in \Gamma^*$.

Дополним такой автомат выходной лентой и выходным алфавитом Δ. Обяжем команду писать $y \in \Delta^*$ на выходной ленте. Полученное устройство называют *(расширенным) преобразователем с магазинной памятью*, подразумевая преобразование входной цепочки в выходную.

Команда преобразователя M будет иметь вид

$$(q_1, a, \gamma_1) \to (q_2, y, \gamma_2).$$

Слова "u есть подцепочка цепочки z"будем выражать записью "$u \subseteq z$".

Цепочка $x \in \chi(G)$ представляется в виде

$$z_1 y_1 \dots z_k y_k,$$

где для $1 \le i \le k$ $y_i \in \Sigma \cup \{\bot\} \cup \{\rangle_p | 0 \le p \le s\}$, а z_i может содержать только роли и открывающие скобки множества $\{\langle_p |\ 0 \le p \le s\}$.

Прогнозы

$$Z_i = \{r \in \mathcal{R} \mid r \subseteq z_i\},\ 1 < i \le k,$$

будут одновременно состояниями и магазинными символами преобразователя. Такой прогноз содержит один или два элемента. Последнее возможно, если $z_i y_i = r\beta\langle_p r'\rangle_p$, где

$$\beta \in \{\langle_q |\ 1 \le q \le s\}^*,$$

а r' — роль пустой правой части pго правила. Тогда

$$Z_i = \{r, r'\}.$$

Обозначим через K_x множество

$$\{Z_i | 1 < i \le k\}$$

всех состояний, определяемых цепочкой x. Пусть

$$K = \cup_{x \in \chi(G)} K_x.$$

Определим преобразователь $M(G) = (\{Z_0, f, \perp\} \cup K \cup \Sigma \cup \Sigma \times \{p | 0 \le p \le s\}, \Sigma \cup \{\perp\}, \{Z_0\} \cup K, Z_0, \{p \mid 0 \le p \le s\}, \cup_{x \in \chi(G)} \delta_x, Z_0, \{f\})$.

Сформулируем правила построения подмножества δ_x команд преобразователя, заметив, что всегда

$$z_1 y_1 = \langle_0 \perp,$$

Z_2 содержит роль $[0, 1, \Lambda]$ и

$$z_{k-1} y_{k-1} z_k y_k = [0, 2, \Lambda]\perp[0, 3, \Lambda]\rangle_0.$$

Началу $z_1 y_1 z_2$ цепочки x сопоставляется команда

(1) $(Z_0, \perp, Z_0) \to (Z_2, \Lambda, Z_0 Z_2)$.

Окончанию $[0, 2, \Lambda]\perp[0, 3, \Lambda]\rangle_0$ цепочки x сопоставляется команда

(2) $(\perp, \Lambda, Z_0 Z_2\{[0, 2, \Lambda]\}) \to (f, 0, Z_0)$.

Остальные команды из δ_x определяются следующими формулами (3)–(6), в которых Z — прогноз, составленный ролями цепочки $z \in \{z_i | 1 < i \le k\}$, и аналогично определяется Z со штрихом или индексом.

(3) $(q, b, Z) \to (Z', \Lambda, ZZ')$, $zaz' \subseteq x$, $a \ne \perp$, $(q, b) \in \{(Z, a), (a, \Lambda)\}$.

(4) $(q, b, Z) \to (a, p, ZZ')$, $z\rangle_p z' \subseteq x$, $[p, 0, a] \in Z$, $(q, b) \in \{(Z, a), (a, \Lambda)\}$.

Следующие формулы (5)–(6) обусловлены соотношениями

$zy^{(1)}z^{(1)} \dots y^{(n_p)}z^{(n_p)}\rangle_p z' \subseteq x$, $n_p > 0$,

для $1 \le j \le n_p$ $[p, j, a] \in Z^{(j)}$, $z = u\langle_p v$, цепочка $vy^{(1)}z^{(1)} \dots y^{(n_p)}z^{(n_p)}$ сбалансирована, $(q, b) \in \{(Z^{(n_p)}, a), (a, \Lambda)\}$.

(5) $(q, b, ZZ^{(1)} \dots Z^{(n_p)}) \to ((a, p), \Lambda, Z)$.

(6) $((a, p), \Lambda, Z) \to (a, p, ZZ')$.

Проводя аналогию с анализатором типа "сдвигєвертка"[Aho -Ullman 72], можно сказать, что команды вида (1) и (3) выполняют сдвиг, а команда (4) и последовательно выполняемые команды (5), (6) выполняют свертки. Команда (2) реализует действие "стоп", но можно также сказать, что она выполняет свертку по нулевому правилу грамматики $Augm(G)$.

Для каждого прочитанного входного символа в магазин преобразователя $M(G)$ поступает прогноз, содержащий роль этого символа. Формулы (5), (6) показывают, что состояние из $\Sigma \cup \{\perp\}$ "запоминает использованный для свертки"входной символ до момента засылки в магазин прогноза, содержащего роль этого символа.

Доопределим функцию $reduction$ на некоторых разбиениях цепочек множества $\{\rho(\sigma)|\sigma \in \mathcal{S}_G\}$, полагая

$$reduction(\rho(u_1, \ldots, u_m)) = \{\rho(x)|x \in reduction(u_1, \ldots, u_m)\}.$$

Тогда наша обычная техника редуцирования позволяет доказать следующее утверждение.

Лемма 13. Пусть $\sigma \in \mathcal{S}_G$ — терминальная структура и δ_σ — множество команд, построенных по $\rho(\sigma)$ по данным выше правилам (1)–(6). Тогда δ_σ входит в множество команд преобразователя $M(G)$ □

Понятия дуги, маршрута и т. пр., введенные для магазинного автомата, очевидным образом обобщаются на случай преобразователя с магазинной памятью. Это дает нам право применять введенные в главе 1 термины и обозначения к преобразователю $M(G)$.

Обсудим особенности маршрутов преобразователя $M(G)$. В этом обсуждении будет полезным

Пусть $G = (N, \Sigma, P, S)$ — бесконтекстная грамматика, D — дерево вывода некоторой цепочки языка $L(G)$. Тогда назовем (D, j)-*кроной* крону ([Aho - Ullman 72], стр. 165) дерева, полученного из D удалением всех вершин, уровень которых больше j.

Заметим, что (D, j)крона является сентенциальной формой, не обязательно правовыводимой.

Только команда вида (2) может перевести преобразователь $M(G)$ в заключительное состояние. Следовательно, его успешный маршрут нейтрален и имеет вид

(7) $\pi_0 T_0 \pi \pi'$,

где дуги π_0, π и π' определяются командами вида (1), (6) и (2) соответственно, T_0 есть нейтральный маршрут.

Из вида команд преобразователя $M(G)$ следует также

Лемма 14. Каждый собственный непустой нейтральный участок успешного маршрута преобразователя $M(G)$ имеет вид

(8) $T_1 \pi_1 \ldots T_k \pi_k \pi_{k+1}$,

где $k \geq 1$, для $1 \leq i \leq k$ T_i является нейтральным маршрутом, дуга π_i определяется командой какого-либо из видов (3), (4), (6), π_{k+1} — командой вида (5). Если следующая за π_{k+1} дуга пишет на выходной ленте номер p и $\mu(\pi_{k+1}) = -\gamma$, то $\gamma = \rho(X_{p1} \ldots X_{pn_p})$ □

Назовем дуги $\pi_1, \ldots, \pi_k$ в разложении (8) собственного нейтрального участка успешного маршрута преобразователя $M(G)$ *каркасными дугами* данного участка. Назовем дуги π_0, π в разложении (7) *каркасными дугами данного успешного маршрута*.

Пусть T — успешный маршрут преобразователя $M(G)$. Определим рекурсивно *уровень каркасных дуг* нейтральных участков маршрута T:

1) каркасные дуги маршрута T — это дуги уровня 1;

2) пусть $j > 1$ и T' — такой непустой нейтральный участок маршрута T, что непосредственно за ним в T следует дуга уровня $j - 1$; тогда назовем каркасные дуги участка T' дугами уровня j.

Пусть $X \in V \cup \{\perp\}$, $a, b \in \Sigma \cup \{\perp\}$, $r = (p, j) \in positions(X)$. Пусть Z есть прогноз, состоящий либо из одной роли $[p, j, a]$, либо из двух ролей $[p, j, a]$ и $[q, 0, b]$, где $(q, 0) \in positions(\Lambda)$. Тогда полагаем $symb(Z) = symb(r)$.

Пусть T — успешный маршрут преобразователя $M(G)$. Пусть для некоторого $k \geq 1$ $T = T_{j0}\pi_{j1}T_{j1} \ldots \pi_{jk}T_{jk}$, где $\pi_{j1}, \ldots, \pi_{jk}$ есть все дуги маршрута T, уровень которых не превосходит j. Тогда назовем (T, j)-*формой* цепочку

$$form(T, j) = symb(Z_{j1}) \ldots symb(Z_{jk})\perp,$$

где $\mu(\pi_{ji}) = +Z_{ji}$ для $1 \leq i \leq k$.

Заметим, что если m — максимальный из возможных для данного успешного маршрута T уровней, то для любой дуги уровня m ее заряд $+Z$ удовлетворяет условию $symb(Z) \in \Sigma \cup \{\perp\}$. Более того, верна

Лемма 15. Если $form(T, j)$ не содержит нетерминальных символов, то

$$form(T, j) = \omega(T)$$

□

Лемма 16. Пусть успешный маршрут T преобразователя $M(G)$ имеет (T, j)-форму для некоторого $j \geq 1$. Тогда она является сентенциальной формой (вообще говоря, неправовыводимой) грамматики $Augm(G)$. Следовательно, $\omega(T) \in L(Augm(G))$.

Доказательство. Достаточно доказать, что (T, j)форма является (D, j)кроной некоторого дерева D вывода в грамматике $Augm(G)$.

Применим индукцию по номеру уровня. Для любого успешного маршрута T $(T, 1)$форма есть $\perp S\perp$, т. е. $(D, 1)$крона любого дерева D вывода из начального символа S_0 грамматики $Augm(G)$. Пусть $j > 1$. Предположим, что (T, i)форма является (D, i)кроной для всех $1 \leq i < j$. Теперь заметим, что (T, j)форма получается из $(T, j-1)$формы заменой каждого нетерминального символа, отвечающего некоторой дуге уровня $j-1$, правой частью правила, номер которого эта дуга пишет на выходную ленту (ср. лемму 14). Но тогда из предположения индукции следует, что (T, j)форма является (D', j)кроной, где деревья D и D' совпадают до $(j-1)$го уровня. Из доказанного и леммы 15 действительно следует, что пометка успешного маршрута принадлежит языку $L(Augm(G))$ □

Будем говорить, что преобразователь $\mathcal{M}$ допускает входную цепочку, если она допускается магазинным автоматом, согласованным с этим преобразователем. Множество всех допускаемых цепочек обозначим через $L(\mathcal{M})$.

Теорема 6. $L(M(G)) = L(Augm(G))$. Для любой цепочки $x \in L(Augm(G))$ множество

$$\{\Pi \mid (Z_0, x, Z_0, \Lambda) \vdash^* (f, \Lambda, Z_0, \Pi)\}$$

содержит все обращения правых выводов цепочки x.

Доказательство. Из леммы 13 следует, что $M(G)$ допускает каждую цепочку из $L(Augm(G))$. Лемма 16 означает, что допускаются только они. Из указанного в доказательстве леммы 16 закона перехода от $(T, j-1)$формы к (T, j)форме следует вторая часть теоремы □

2.2.2. Уменьшение степени недетерминированности преобразователя $M(G)$. Далее вместо пары команд (5), (6) будем рассматривать одну:

(9) $(q, b, ZZ^{(1)} \dots Z^{(n_p)}) \to (a, p, ZZ')$.

Так измененный преобразователь по-прежнему называем $M(G)$. Теперь преобразуем $M(G)$ так, чтобы недетерминированность в его поведении уменьшилась. Обозначим новый преобразователь через $ELRM(G)$. Его команда будет определяться некоторым множеством команд преобразователя $M(G)$. Состояниями (и магазинными символами) станут некоторые объединения прогнозов, являющихся состояниями и магазинными символами преобразователя $M(G)$. Начальное и заключительное состояния и маркер дна магазина наследуются от $M(G)$.

Пусть v — прогноз. Пусть $\pi_i = [p_i, j_i, a_i] \in v$, $i = 1, 2$. Пусть выполняется какое-либо из следующих условий:

1) $j_1 = n_{p_1}$, $j_2 = n_{p_2}$ и $p_1 \neq p_2$, $a_1 = a_2$,

2) $j_1 = n_{p_1}$, $j_2 < n_{p_2}$ и

$$\exists x \in (V \cup \{\perp\})^* : \ X_{p_2(j_2+1)} \dots X_{p_2 n_{p_2}} a_2 \Rightarrow^*_{Augm(G)} a_1 x.$$

Тогда будем говорить, что π_1 и π_2 *конфликтуют* (друг с другом) в v.

Обозначим через $NoConfl(v)$ совокупность Set подмножеств прогноза v, получаемую действиями:

$Set := \{v\}$,

пока Set содержит множество v', элементы π_1 и π_2 которого в нем конфликтуют, **выполнять** присваивание

$$Set := Set - \{v'\} \cup \{v' - \{\pi_1\}, v' - \{\pi_2\}\}.$$

Так как любой шаг приведенного построения уменьшает мощности множеств, содержащих конфликтующие элементы, то для любого прогноза v (он конечен в силу конечности грамматики) множество $NoConfl(v)$ существует и его мощность не превосходит $\mid v \mid$.

Теперь приведем формулы, которые позволяют построить все состояния, магазинные символы и команды преобразователя $ELRM(G)$, располагая только его начальным состоянием Z_0 и командами преобразователя $M(G)$.

Пусть $0 \leq p \leq s$, пусть v и $v^{(j)}$, $0 \leq j \leq n_p$, — состояния (и магазинные символы) преобразователя $ELRM(G)$, $a \in \Sigma \cup \{\perp\}$. Введем обозначения:

$$vert(v, a) = NoConfl(\{r \in Z' \mid (q, b, Z) \to (Z', \Lambda, ZZ') \in \delta_{M(G)},$$
$$Z \subseteq v, \ (q, b) \in \{(Z, a), (a, \Lambda)\}\})$$

и

$$vert(v^{(0)}, \dots, v^{(n_p)}, p, a) = NoConfl(\{r \in Z' \mid$$
$$(q, b, Z^{(0)} Z^{(1)} \dots Z^{(n_p)}) \to (a, p, Z^{(0)} Z') \in \delta_{M(G)};$$
$$\text{для } 0 \leq j \leq n_p \ Z^{(j)} \subseteq v_j; \ (q, b) \in \{(Z^{(n_p)}, a), (a, \Lambda)\}\}).$$

Тогда $ELRM(G)$ имеет команды:

(10) $(q, b, v) \to (v', \Lambda, vv')$, $v' \in vert(v, a)$, $(q, b) \in \{(v, a), (a, \Lambda)\}$,

(11) $(q, b, v^{(0)} v^{(1)} \dots v^{(n_p)}) \to (a, p, v^{(0)} v')$, $p > 0$,
$v' \in vert(v^{(0)}, \dots, v^{(n_p)}, p, a)$, $(q, b) \in \{(v^{(n_p)}, a), (a, \Lambda)\}$.

(12) $(\perp, \Lambda, Z_0 v^{(1)} v^{(2)}) \to (f, 0, Z_0)$, $[0, 1, \Lambda] \in v^{(1)}$, $[0, 2, \Lambda] \in v^{(2)}$.

Пример 2. Для грамматики G с правилами

1. $S \to aSb$
2. $S \to ab$

$ELRM(G)$ имеет команды:

1) $(Z_0, \perp, Z_0) \to (v_0, \Lambda, Z_0 v_0)$, $v_0 = \{01\Lambda\}$;
2) $(v_0, a, v_0) \to (v_2, \Lambda, v_0 v_2)$, $v_2 = \{11\perp, 21\perp\}$;
3) $(v_2, b, v_2) \to (v_5, \Lambda, v_2 v_5)$, $v_5 = \{22\perp\}$;
4) $(v_2, a, v_2) \to (v_4, \Lambda, v_2 v_4)$, $v_4 = \{11b, 21b\}$;
5) $(v_5, \perp, v_0 v_2 v_5) \to (\perp, 2, v_0 v_1)$, $v_1 = \{02\Lambda\}$;
6) $(v_4, b, v_4) \to (v_8, \Lambda, v_4 v_8)$, $v_8 = \{22b\}$;
7) $(v_4, a, v_4) \to (v_4, \Lambda, v_4 v_4)$;
8) $(v_8, b, v_2 v_4 v_8) \to (b, 2, v_2 v_3)$, $v_3 = \{12\perp\}$;
9) $(v_8, b, v_4 v_4 v_8) \to (b, 2, v_4 v_7)$, $v_7 = \{12b\}$;
10) $(b, \Lambda, v_3) \to (v_6, \Lambda, v_3 v_6)$, $v_6 = \{13\perp\}$;
11) $(v_6, \perp, v_0 v_2 v_3 v_6) \to (\perp, 1, v_0 v_1)$;
12) $(b, \Lambda, v_7) \to (v_9, \Lambda, v_7 v_9)$, $v_9 = \{13b\}$;
13) $(v_9, b, v_2 v_4 v_7 v_9) \to (b, 1, v_2 v_3)$;
14) $(v_9, b, v_4 v_4 v_7 v_9) \to (b, 1, v_4 v_7)$;
15) $(\perp, \Lambda, Z_0 v_0 v_1) \to (f, 0, Z_0)$.

Данный преобразователь является детерминированным.

Будем называть эквивалентными преобразователи, которые допускают одинаковые множества входных цепочек, одинаково преобразуя каждую входную цепочку.

С помощью формул (10)–(12) нетрудно доказать индукцией по длине маршрута, что $M(G)$ определяет маршрут T с начальной вершиной (Z_0, Z_0) и конечной вершиной (q, Z) тогда и только тогда, когда $ELRM(G)$ определяет маршрут T' с теми же длиной, начальной вершиной, пометкой и выходом и конечной вершиной (v, Z), где либо $v = q \in \Sigma \cup \{\perp\}$, либо v — прогноз, содержащий q. Следовательно, $M(G)$ и $ELRM(G)$ эквивалентны.

2.2.3. Построение по $ELRM(G)$ недетерминированного слеванаправо анализатора. Как и $M(G)$, $ELRM(G)$ генерирует каждый разбор входной цепочки. Таким образом, $ELRM(G)$ является недетерминированным анализатором. Придадим ему более обычную в теории синтаксического анализа форму таблиц, управляющих работой алгоритма анализа. Определим две таблицы анализа слева направо без возвратов и с заглядыванием вперед на один входной символ, близкие по организации к введенным в классической работе [Knuth 65]. Одну назовем таблицей действий и обозначим через $AT(G)$, другую назовем таблицей переобозначений и обозначим через $RT(G)$. Их строчки имеют соответственно вид (прогноз, входной символ, действие) и (прогноз, символ из $V - \{\perp\}$, прогноз). Действие есть сдвиг или свертка, задаваемая номером правила, которое необходимо применить. Будем считать, что сдвиг маркера конца входной цепочки вызывает останов анализатора.

Название таблицы переобозначений связано с тем, что каждый прогноз анализатора можно интерпретировать как новое обозначение некоторого символа грамматики $Augm(G)$. Возможность такой интерпретации прогнозов позволяет хранить в магазине анализатора одни прогнозы, но мы, следуя сложившейся традиции, будем считать, что в магазине хранятся пары вида (символ, прогноз).

Правила построения таблиц по преобразователю $ELRM(G)$ таковы. Команда вида (10) определяет строчки

$$(v, a, v') \in RT(G) \text{ и } (v, a,) \in AT(G).$$

Команда вида (11) определяет строчки

$$(v^{(0)}, A_p, v') \in RT(G) \text{ и } (v^{(n_p)}, a, p) \in AT(G).$$

Команда вида (12) определяет строчки

$$(v^{(1)}, S, v^{(2)}) \in RT(G) \text{ и } (v^{(2)}, \perp, \text{сдвиг}) \in AT(G).$$

Работу самого анализатора (назовем его ELR(1)анализатором, подразумевая обобщение LR(1)техники, и обозначим через $\mathcal{A}(G)$) можно описать следующим образом.

Алгоритм 4. Один из вариантов обработки входной цепочки, которые имеет право выбрать недетерминированный ELR(1)анализатор $\mathcal{A}(G)$.

Вход. $AT(G)$; $RT(G)$; цепочка x, подлежащая анализу.

Выход. Последовательность П, представляющая правый вывод цепочки x в грамматике G, или сигнал неудачи.

Вспомогательные обозначения: $input$ — очередной входной символ, $next$ — символ, подлежащий засылке в магазин, top — верхний в магазине прогноз.

Метод. **Пока** не сдвигался маркер конца входной цепочки, **выполнять**:
если $\exists$ *действие*: $(top, input,$ *действие*$) \in AT(G)$,
то
если *действие=сдвиг*,
то ($next := input$, продвинуться на одну позицию во входной цепочке);
если *действие=p*,
то ($next := A_p$; удалить из магазина n_p элементов, добавить в последовательность П номер p);
если $\exists \pi : (top, next, \pi) \in RT(G)$,
то заслать в магазин пару $(next, \pi)$,
иначе сигнализировать неудачу
иначе сигнализировать неудачу.

Из конструкции данного алгоритма и его таблиц следует корректность $\mathcal{A}(G)$ в том смысле, что он заканчивает удачно для цепочек из

$$\perp L(G) \perp = L(Augm(G))$$

и только для них, формируя при этом разбор цепочки. Буква E в термине "ELR(1)-анализатор"указывает на обобщение техники LR(1)анализа на случай любой приведенной бесконтекстной грамматики.

Как видно, алгоритм $\mathcal{A}(G)$ совпадает с LR(1)алгоритмом, если пара $(top, input)$ однозначно определяет действие, а $(top, next)$ однозначно определяет прогноз. Другими словами, если преобразователь $ELRM(G)$ является детерминированным, то $\mathcal{A}(G)$ осуществляет LR(1)анализ. Нетрудно указать и переход от таблиц LR(1)анализа к магазинному преобразователю, совпадающему с $ELRM(G)$. Таким образом, получаем следующее важное утверждение.

Теорема 7. Грамматика G является LR(1)ŗграмматикой тогда и только тогда, когда $ELRM(G)$ — детерминированный преобразователь □

2.3. Некоторые приемы уменьшения анализаторов

2.3.1. Уменьшение анализатора, имеющего форму преобразователя. $ELRM(G)$ использует магазин явно чрезмерно, "пропуская"через него обозначение каждого прочитанного символа и каждого символа, полученного сверткой. В то же время в командах этого преобразователя состояния частично или полностью выражают информацию, черпаемую из магазина. Попытаемся модифицировать $ELRM(G)$ так, чтобы магазин не использовался в тех случаях, когда достаточно состояний.

Ясно, что магазин необходим при наличии в грамматике самовставляющихся символов. Но иногда он нужен и в случае грамматики без самовставлений, чтобы правильно сформировать выход.

Далее $\mathcal{M}$ обозначает преобразователь $ELRM(G)$ или результат некоторого его преобразования.

Пусть длины маршрутов T и T' равны. Пусть эти маршруты прочитывают одинаковые входные цепочки и пишут одинаковые выходные. Пусть для любых T_i и T'_i, $i = 1, 2, 3$, таких, что маршруты с одним номером имеют одну длину и верны равенства $T = T_1T_2T_3$ и $T' = T'_1T'_2T'_3$, T_i и T'_i прочитывают одинаковые входные цепочки и формируют одинаковые выходные. Тогда будем говорить, что T проецируется на T' и при этом T'_2 является проекцией маршрута T_2, а вершина $beg(T'_2)$ является проекцией вершины $beg(T_2)$.

Пусть автомат $\mathcal{M}$ получен из автомата $\mathcal{M}'$ некоторым преобразованием. Пусть существует такая биекция

$$\phi : \ Core(\mathcal{M}) \to Core(\mathcal{M}'),$$

что $T \in Core(\mathcal{M})$ проецируется на $\phi(T)$, причем родословные маршрутов T и $\phi(T)$ с элементарными предками имеют одинаковые длины и T_j проецируется на T'_j, $1 \le j \le 5$, для тех элементов $(T_1, T_2, T_3, T_4, T_5)$ и $(T'_1, T'_2, T'_3, T'_4, T'_5)$ этих родословных, которые имеют в них один и тот же порядковый номер. Тогда назовем преобразование автомата $\mathcal{M}$ в $\mathcal{M}'$ *сохраняющим родословные.*

Из теоремы 1.4 (о развитии) следует

Лемма 17. Пусть преобразование $\mathcal{M}$ в $\mathcal{M}'$ сохраняет родословные. Тогда $\mathcal{M}$ и $\mathcal{M}'$ эквивалентны □

Заметим, что для сохраняемости родословных достаточно, чтобы проекции различных вершин были различны, проекции цикла и нецикла были соответственно циклом и нециклом, проекции парных циклов также были парными циклами. Однако, эти условия не необходимы, как можно убедиться на примере магазинного автомата, определяемого командами

$(p_0, a, Z_0) \to (p_1, Z_0a)$,
$(p_1, a, a) \to (p_1, aa)$,
$(p_1, b, a) \to (p_2, \Lambda)$,
$(p_2, b, a) \to (p_3, \Lambda)$,
$(p_3, b, Z_0) \to (f, Z_0)$.

Единственный успешный маршрут данного автомата является элементарным. Автомат с командами

$(p_0, a, Z_0) \to (p_1, Z_0)$,
$(p_1, a, Z_0) \to (p_4, Z_0)$,
$(p_4, b, Z_0) \to (p_2, Z_0)$,
$(p_2, b, Z_0) \to (p_3, Z_0)$,
$(p_3, b, Z_0) \to (f, Z_0)$

эквивалентен предыдущему, но, в отличие от него, не имеет циклов.

Пусть $\epsilon(\gamma, v)$, где γ — цепочка магазинных символов, обозначает результат вычеркивания из γ всех вхождений символа v.

Пусть $v \neq Z_0$ — магазинный символ автомата $\mathcal{M}$. Обозначим через $\epsilon(\mathcal{M}, v)$ автомат, получаемый из $\mathcal{M}$ выполнением следующих действий:

каждую команду вида
$(q_1, b, v\gamma_1) \to (q_2, y, \gamma_2)$
заменить командами множества
$\{(q_1, b, v'\gamma_1) \to (q_2, y, v'\gamma_2) | v' \neq v, \exists$ команда вида $(q, a, v'\gamma) \to (q', y', v'v)\}$ (из конструкции преобразователя $ELRM(G)$ и неравенства $v \neq Z_0$ следует, что если такая замена требуется, то она всегда осуществима);

каждую команду
$(q, a, \gamma) \to (q', y, \gamma')$
заменить командой
$(q, a, \epsilon(\gamma, v)) \to (q', y, \epsilon(\gamma', v))$.

Заметим, что такое преобразование может привести к появлению команд, которые не влияют на содержимое магазина или же заменяют пустой цепочкой суффикс обозреваемой магазинной цепочки.

Пусть q, q', b, γ и y таковы, что для любой команды вида $(q, b, \gamma_1) \to (q', y, \gamma_2)$ существует такая непустая магазинная цепочка γ', что $\gamma_1 = \gamma'\gamma$, $\gamma_2 = \gamma'$. Тогда назовем (q, q', b, γ, y)-группой множество
$\{\theta \mid \exists \gamma' \in \Gamma^+ : \theta = (q, b, \gamma'\gamma) \to (q', y, \gamma')\}$
команд данного преобразователя. (Содержательно, переход, определяемый любой командой группы, не зависит от префикса γ' обозреваемой магазинной цепочки $\gamma'\gamma$.) В противном случае (q, q', b, γ, y)-группа пуста.

Назовем *законным* каждое из следующих двух преобразований автомата $\mathcal{M}$: сохраняющее родословные преобразование $\mathcal{M}$ в $\epsilon(\mathcal{M}, v)$ и замену непустой (q, q', b, γ, y)-группы одной командой $(q, b, \gamma) \to (q', y, \Lambda)$.

Алгоритм 5. Уменьшение преобразователя.

Вход. Преобразователь $\mathcal{M}$.

Выход. Наилучший (по числу команд или по каким-либо другим свойствам) из преобразователей, которые получаются следующими преобразованиями автомата $\mathcal{M}$.

Шаг 1. Построить по $\mathcal{M}$ преобразователь $\mathcal{M}_1$ — результат некоторой последовательности законных преобразований, к которому уже не применимы законные преобразования.

Шаг 2. Если полученный шагом 1 преобразователь $\mathcal{M}_1$ является детерминированным, то применить к нему модификацию алгоритма 3, устраняющую команды,

которые не читают входного символа, не пишут выходных символов и не стирают в магазине. (Нужная модификация достигается переопределением множества дуг, которые подлежат устранению: теперь они должны быть не только нечитающими и нестирающими, но и не писать на выходную ленту.)

Нетрудно убедиться, что шаг 1 сохраняет свойство детерминированности. То же доказано для алгоритма, упомянутого шагом 2. Следовательно, алгоритм 2 можно считать и алгоритмом уменьшения детерминированных ELR(1)анализаторов.

2.3.2. Уменьшение числа прогнозов. Заметим, что состояния и магазинные символы преобразователей $M(G)$ и $ELRM(G)$ обозначают вхождения символов грамматики $Augm(G)$ в ее правила. Отсюда следует, что нижняя граница действительно необходимого ELR(1)анализатору числа прогнозов может быть (по крайней мере для некоторых подклассов грамматик) близкой к числу символов грамматики $Augm(G)$.

Следующий алгоритм указывает очевидно разумные попытки уменьшить число прогнозов анализатора $\mathcal{A}(G)$ путем объединения некоторых прогнозов.

Пусть v_1 и v_2 — различные прогнозы, используемые преобразователем $\mathcal{M}$ в качестве состояний (и, возможно, магазинных символов). Если замена каждого использования этих прогнозов объединением $v_1 \cup v_2$ сохраняет родословные, то назовем прогнозы v_1 и v_2 *объединимыми* (друг с другом).

Алгоритм 6. Сокращение числа прогнозов.

Вход и выход. Pr — множество прогнозов; первоначально это прогнозы анализатора $\mathcal{A}(G)$, по окончании — прогнозы сокращенного анализатора.

Метод. **Пока** множество Pr содержит некоторые объединимые прогнозы v_1 и v_2, **выполнять**:

($Pr := Pr - \{v_1, v_2\} \cup \{v_1 \cup v_2\}$;

заменить на $v_1 \cup v_2$ каждое вхождение v_1 и v_2 в команды преобразователя $\mathcal{M}$ (можно вместо этого изменять соответствующие автомату $\mathcal{M}$ таблицы действий и переобозначений).

Алгоритм 3 можно рассматривать и как алгоритм сокращения таблиц анализатора, и как алгоритм сокращения распознавателя.

Заметим, что для одного из тех элементов множества Pr, которые указывают роли нетерминального символа A, можно использовать обозначение A. Тогда сократится общее число обозначений, нужных для описания анализатора. Кроме того, исходная грамматика будет легче "проглядываться"в анализаторе. Такой же прием возможен и для терминальных символов, не являющихся полезными состояниями преобразователя $M(G)$.

Проверка следующих условий резко сокращает число прогнозов, объединимость которых имеет смысл исследовать:

$NoConfl(v_1 \cup v_2) = \{v_1 \cup v_2\}$;

$\forall X \in V \quad ([\ i = 1, 2\ \exists v_{Xi} :\ (v_i, X, v_{Xi}) \in RT(G)] \Longrightarrow v_{X1}$ и v_{X2} объединимы).

Заметим, что если ограничиться распознаванием цепочек и не формировать выход, то несложные модификации алгоритмов 5 и 6 будут производить по преобразователю $ELRM(G)$ эквивалентный магазинный автомат, экономность которого определяется в основном экономностью исходной грамматики G. Модификация

алгоритма 6 для случая магазинных автоматов сможет объединять больше прогнозов, так как отпадает надобность в выходной цепочке.

Порядок применения алгоритмов 5 и 6 не безразличен: от него существенно зависит вид результирующего преобразователя.

Пример 3. Рассмотрим грамматику G с единственным правилом

$$S \to aa$$

Элемент характеристики $\chi(G)$ также один:

$$\langle_0 \bot[01\Lambda]\langle_1 a[11\bot]a[12\bot]\rangle_1[02\Lambda]\bot[03\Lambda]\rangle_0$$

$M(G)$ и $ELRM(G)$ совпадают. Если занумеровать прогнозы в цепочке выше начиная с нуля, то команды преобразователей таковы:

1. $(Z_0, \bot, Z_0) \to (0, \Lambda, Z_0 0)$
2. $(0, a, 0) \to (1, \Lambda, 01)$
3. $(1, a, 1) \to (2, \Lambda, 12)$
4. $(2, \bot, 012) \to (\bot, 1, 03)$
5. $(\bot, \Lambda, Z_0 03) \to (f, 0, Z_0)$

Если алгоритм 5 применить первым к данному преобразователю, то последний превратится в неуменьшаемый конечный преобразователь. Если же применить первым алгоритм 6, то число прогнозов уменьшится, так как прогнозы 1 и 2, указывающие роли символа a, объединимы. После этого объединения изменятся команды 3 и 4:

$3'$. $(1, a, 1) \to (1, \Lambda, 11)$

$4'$. $(1, \bot, 011) \to (\bot, 1, 03)$

Теперь алгоритм 5 способен ликвидировать только магазинные символы 0 и 3, но не 1. Результирующий преобразователь останется магазинным:

1. $(Z_0, \bot, Z_0) \to (0, \Lambda, Z_0)$
2. $(0, a, Z_0) \to (1, \Lambda, Z_0 1)$
3. $(1, a, 1) \to (1, \Lambda, 11)$
4. $(1, \bot, Z_0 11) \to (\bot, 1, Z_0)$
5. $(\bot, \Lambda, Z_0) \to (f, 0, Z_0)$

В детерминированном случае алгоритмы 5 и 6 можно интерпретировать как алгоритмы уменьшения LR(1)анализаторов. Вместе с приемом расщепления [Игнатов 87б] грамматики и приведенными выше необходимыми условиями объединимости прогнозов они дают приемлемый на практике метод построения LR(1)-анализаторов. Построенные таким образом LR(1)анализаторы будут, повидимому, неулучшаемыми.

§3. Прием разделения в обработке языковых описаний

Применяемые на практике модели алгоритмических языков используют зачастую простейшие из типов формальных языковых описаний, зато требуют большого изобретательства при реализации того, что не охватывается этими описаниями. Привлечение более сложных формализмов сокращает область изобретательства, но сталкивается с трудностями их реализации.

Основная трудность — разрастание объема исходного языкового описания при попытке выяснить взаимоотношения его элементов. Идея обрабатывать языковое описание по частям естественна, и примеры ее воплощения возникли давно. Но эти разработки базируются на явлениях, причина которых не объясняется авторами. Механизм разделения на части остается таинственным; не ясно, в чем и насколько радикально можно его совершенствовать.

В данной работе, по существу, сделана попытка объяснить такие механизмы. Повидимому, попытку можно считать удачной, хотя и не совсем законченной. Успех во многом обусловлен графической интерпретацией традиционных синтаксических описаний бесконтекстных языков. Незаконченность состоит в том, что не предложен способ разделения сильно связных компонент Dғрафов, а они могут быть очень велики.

3.1. Построение регулярного выражения по конечному автомату. Регулярные языки играют большую роль в математических теориях и в приложениях. Обширные библиографии по этому поводу имеются во множестве легко доступных источников. Упомянем здесь только работу [Мартыненко 98], которая посвящена методике и технологии синтаксически управляемой обработки данных и в которой используются понятия регулярного выражения и конечного автомата.

Работа [Мартыненко 98] наряду с некоторыми другими убеждает в значимости конструкций, доказывающих теорему Клини об эквивалентности регулярных выражений и конечных автоматов. В нашей работе дается построение регулярного выражения по конечному автомату, которое опирается на эффективные алгоритмы на графах. В этом отношении наша работа, возможно, уникальна, так как подобные разработки нам не известны.

Предлагаемое здесь (см. также [Stanevichene -Vylitok 2000]) построение рекурсивно, как и построение, принадлежащее Клини, но представляется нам легче воспринимаемым. В самом деле, в нашем алгоритме автомат трактуется как достаточно естественная система подавтоматов, тогда как алгоритм Клини базируется на произвольном упорядочении состояний (иначе, вершин) и на связанной с ним весьма формальной классификации путей. Существуют и другие доказательства теоремы Клини; одно из последних содержится в [Melnikov -Vakhitova 98]. Но они также не указывают естественного способа расчленения автомата и не позволяют привлечь испытанные приемы обработки графов.

Наше построение использует для выделения сильно связных компонент (это одна из основных частей построения) алгоритм из [Рейнгольд -Нивергельт -Део 80].

В разделе 3.1.1 настоящей работы вводятся используемые здесь обозначения. В разделе 3.1.2 с помощью понятия сильно связной компоненты определяется иерархия подавтоматов конечного автомата, позволяющая свести его обработку к обработке некоторых меньших его частей. В разделе 3.1.3 формулируется алгоритм построения по конечному автомату регулярного выражения. При этом преследуется цель облегчить доказательство эквивалентности выражениярезультата исходному автомату.

3.1.1. Определение конечного автомата и регулярного выражения. Введем обозначения и термины, которые связаны с понятием конечного автомата и используются в следующих разделах работы.

Конечный автомат (*над алфавитом* Σ) есть пятерка

$$\mathcal{A} = (K,\ \Sigma,\ \delta,\ p_0,\ F),$$

где:

K — конечное множество *состояний*, или *вершин*;

Σ — *входной алфавит* (также конечный);

$\delta \subseteq K \times (\Sigma \cup \{\Lambda\}) \times K$ — множество *команд*, или *дуг*;

p_0 — *начальное состояние*;

$F \subseteq K$ — множество *заключительных состояний*.

Можно говорить о пути на графе $\mathcal{A}$ — *пути конечного автомата* $\mathcal{A}$. Начальную и конечную вершины пути T обозначим через $beg(T)$ и $end(T)$ соответственно.

Пусть T — такой путь, что $beg(T) = p_0$, а $end(T) \in F$. Тогда Т будем называть *успешным путем* или *предложением*. Введем обозначение для множества всех успешных путей конечного автомата $\mathcal{A}$:

$$Sentences(\mathcal{A}) = \{T \mid T \text{ — успешный путь конечного автомата } \mathcal{A}\}.$$

Пусть Σ — некоторый алфавит, не содержащий символов *, +, ϵ, $\emptyset$, (,). Определим рекурсивно *регулярное выражение над алфавитом* Σ (это специфическое слово в алфавите

$$Alph = \Sigma \cup \{*,\ +,\ \epsilon,\ \emptyset,\ (,\ )\ \}) :$$

1) $a \in \Sigma \cup \{\epsilon, \emptyset\}$ является регулярным выражением;

2) если α и β — регулярные выражения, то

а) $\alpha + \beta$ — регулярное выражение; подвыражения α и β данного выражения будем называть его *слагаемыми*, само выражение — *суммой*;

б) $(\alpha)(\beta)$ — регулярное выражение; подвыражения α и β будем называть его *сомножителями*, само выражение — *произведением*; заключать сомножитель в круглые скобки не обязательно, если он не является суммой;

в) $(\beta)^*$ — регулярное выражение, называемое *итерацией* выражения β.

3.1.2. Иерархия в конечном автомате. Напомним необходимые здесь понятия теории графов.

Ориентированный граф *сильно связен*, если для каждых его двух вершин u и v существует путь как из u в v, так и из v в u.

Пусть подграф $G = (V, E)$ некоторого ориентированного графа сильно связен и любой отличный от G подграф (V', E'), такой, что $V \subseteq V', E \subseteq E'$, не является сильно связным. Тогда назовем G *сильно связной компонентой* данного графа.

Связный граф с одной вершиной и пустым множеством дуг будем называть *тривиальным*.

С помощью понятия сильно связной компоненты определим числовую характеристику связного подграфа кончного автомата, которая позволяет указать естественную иерархию некоторых из его частей, рассматриваемых как автоматы. Степень подчиненности подавтомата тем больше, чем ниже соответствующее число — ранг.

Пусть $\mathcal{B}$ — подграф конечного автомата над алфавитом Σ, $C = (V, E)$ — нетривиальная сильно связная компонента графа $\mathcal{B}$, $\pi \in E$. Тогда обозначим конечный автомат $(V, \Sigma, E - \{\pi\}, end(\pi), \{beg(\pi)\})$ через $Finite(\mathcal{B}, \pi)$.

Пусть $\mathcal{C}$ — связный подграф конечного автомата. Определим рекурсивно *ранг* $rank(\mathcal{C})$ этого подграфа:

1) $rank(\mathcal{C})$=0, если $\mathcal{C}$ — тривиальная сильно связная компонента;

2) $rank(\mathcal{C}) = 1 + min\{rank(Finite(\mathcal{C}, \pi)) \mid \pi$ — дуга графа $\mathcal{C}\}$, если граф $\mathcal{C}$ совпадает со своей нетривиальной сильно связной компонентой;

3) $rank(\mathcal{C}) = max\{rank(\mathcal{B}) \mid \mathcal{B}$ — сильно связная компонента графа $\mathcal{C}\}$.

3.1.3. Преобразование конечного автомата в регулярное выражение

Обозначим через $Lines(\mathcal{A})$ множество предложений автомата $\mathcal{A}$, не содержащих циклов:

$$Lines(\mathcal{A}) = \{T \in Sentences(\mathcal{A}) \mid \forall (T_1, T_2, T_3)$$
$$T = T_1 T_2 T_3 \Rightarrow (T_2 \text{ не является циклом})\}.$$

Определим рекурсивно регулярное выражение $\mathcal{E}(A)$ над алфавитом дуг автомата $\mathcal{A}$; при этом каждый пустой путь будем считать синонимом обозначения Λ:

$$\mathcal{E}(A) = \sum_{T \in Lines(\mathcal{A})} Cycles(beg(T)) Addend(T),$$

$$Addend(T) = \begin{cases} \epsilon, \text{ путь } T \text{ пуст}, \\ \pi_1 Cycles(end(\pi_1)) ... \pi_k Cycles(end(\pi_k)), \ k \geq 1, \\ \quad T = \pi_1 ... \pi_k, \ \pi_i \in \delta, \ 1 \leq i \leq k, \end{cases}$$

где вспомогательное выражениеєумма $Cycles(p)$, $p \in K$, определяется в следующем абзаце.

Если p является вершиной нетривиальной сильно связной компоненты, то

$$Cycles(p) = (\sum_{\substack{\pi\text{— дуга сильно связной компоненты,} \\ beg(\pi)=p}} Bound(\pi))^*,$$

иначе

$$Cycles(p) = p;$$

здесь для дуги π сильно связной компоненты $\mathcal{B}$ автомата $\mathcal{A}$

$$Bound(\pi) = \pi(\mathcal{E}(Finite(\mathcal{B}, \pi))).$$

Теорема 8. $L(\mathcal{E}(A)) = Sentences(\mathcal{A})$.

Доказательство. Индукцией по $r = rank(\mathcal{A})$. При $r = 0$ предложения автомата не имеют циклов, т.е. справедливо равенство $Lines(\mathcal{A}) = Sentences(\mathcal{A})$. Следовательно, формула для $\mathcal{E}(A)$ упрощается в данном случае до

$$\mathcal{E}(A) = \sum_{T \in Sentences(\mathcal{A})} T$$

и теорема верна.

Пусть $r > 0$. Предположим, что теорема выполняется для каждого автомата, ранг которого не превышает $r - 1$, и рассмотрим автомат $\mathcal{A}$ ранга r.

Покажем, что для любой сильно связной компоненты $\mathcal{B}$ автомата $\mathcal{A}$ и любой ее вершины p регулярное выражение $Cycles(p)$ задает все циклы с концевой вершиной p. Действительно, пусть π — дуга данной сильно связной компоненты и $beg(\pi) = p$. Рассмотрим слагаемое $Bound(\pi)$, используемое для определения выражения $Cycles(p)$. По предположению индукции его подвыражение $\mathcal{E}(Finite(\mathcal{B}, \pi))$ задает все пути из $end(\pi)$ в $beg(\pi) = p$, не содержащие дуги π. Таким образом, выражение $Bound(\pi)$ задает все циклы с концом p, которые начинаются дугой π и не содержат других ее вхождений. В целом сумма, стоящая под знаком итерации в определении выражения $Cycles(p)$, задает все циклы с концом p, первая дуга которых в них не повторяется. Заметим, что произвольный цикл с концом p — это последовательность таких циклов. Но тогда ясно, что итерация рассматриваемой суммы задает все пути из p в p.

Отсюда следует, что сумма $\mathcal{E}(A)$ задает все пути, начальная вершина которых есть p_0, а конечная принадлежит множеству заключительных вершин автомата $\mathcal{A}$ $\square$

Определение регулярного выражения $\mathcal{E}(A)$ влечет рекурсивный алгоритм преобразования конечного автомата в эквивалентное регулярное выражение.

Пример 4. Рассмотрим детерминированный конечный автомат

$$\mathcal{A} = (\{p_0, p_1, p_2, p_3\},\ \{a, b\},\ \delta,\ p_0,\ \{p_2\}),$$

где δ определяется следующим списком команд:
$(p_0, a) \to p_1$,
$(p_1, a) \to p_2$,
$(p_1, b) \to p_3$,
$(p_3, b) \to p_1$,
$(p_3, a) \to p_3$.

Ранг автомата равен двум, так как его нетривиальная сильно связная компонента $\mathcal{C}$ (ее вершины — p_1, p_3, дуги — p_1bp_3, p_3bp_1, p_3ap_3) имеет ранг 1. Действительно, она содержит нетривиальную сильно связную компоненту и после вычитания любой одной ее дуги.

Множество $Lines(\mathcal{A})$ включает только один путь:

$$p_0ap_1ap_2.$$

Его промежуточная вершина p_1 является вершиной сильно связной компоненты $\mathcal{C}$.

Подвыражение $Cycles(p_1)$ регулярного выражения $\mathcal{E}(A)$ определяется произведением

$$(p_1bp_3)\mathcal{E}(B),$$

где

$$\mathcal{B} = Finite(\mathcal{C},\ p_1bp_3).$$

Нетрудно вычислить $\mathcal{E}(B)$:

$$\mathcal{E}(B) = (p_3ap_3)^*p_3bp_1$$

Следовательно,

$$\mathcal{E}(A) = p_0ap_1(p_1bp_3(p_3ap_3)^*p_3bp_1)^*p_1ap_2.$$

Отображая дуги в их пометки, получаем, что исходному автомату отвечает регулярное выражение

$$a(ba^*b)^*a.$$

3.2. Преобразование Dғрафа в КСвыражение.

Идея обработки формальных синтаксических описаний по частям возникла давно. Наиболее заметные ее воплощения связаны с трудностями реализации LR(k)-анализаторов [Korenjak 69], [Игнатов 87а], [Игнатов 87б]. По сути разработчики компиляторов всегда прибегают к разделению языковых описаний, зачастую отступая от синтаксиса, предложенного создателями языка. Так, синтаксис языка АЛГОЛ68 получил любопытное преломление: рассмотренная в [Мартыненко 98] реализация языка использует систему регулярных выражений, алфавиты которых включают "семантические символы— действия по обработке контекстных условий.

Подходы упомянутых выше работ [Korenjak 69], [Игнатов 87а], [Игнатов 87б] ценны тем, что система частей, заменяющих исходную грамматику, не требует особых ухищрений по прослеживанию связи понятий исходного и производного описаний языка. Алгоритм расщепления грамматики, предложенный в [Korenjak 69], рассчитывает на вмешательство человека; в [Игнатов 87а], [Игнатов 87б] работа по расщеплению полностью автоматизирована. В [Korenjak 69], [Игнатов 87а], [Игнатов 87б] показано, что приближения языков программирования КС-грамматиками хорошо поддаются расщеплению, хотя не каждая КСғрамматика может быть расщеплена на части, допускающие некоторую независимую их обработку, необходимую для создания анализатора.

Здесь предлагаются приемы разделения Dғрафов, облегчающие преобразование Dғрафа в эквивалентное КСвыражение. Так как Dғрафы являются средством задания КСязыков, работа может оказать влияние, в частности, на технологии синтаксически управляемой обработки данных.

Наш метод разделения Dғрафа предполагает применение алгоритмов на графах [Рейнгольд - Нивергельт - Део 80], что является, повидимому, новшеством в технике обработки языковых описаний. Большую роль при этом играет понятие сильно связной компоненты графа.

В разделах 3.2.1–3.2.2 изучается состав и взаимозависимость сильно связных компонент Dғрафа в связи с наблюдаемым в Dғрафе явлением парности дуг. Оказывается, Dғраф может иметь циклы, которые никогда не итерируются в маршрутах, т.е. наряду с маршрутом $T_1T_2T_3$, где T_2 — подобный цикл, не существует маршрутов вида $T_1'T_2^nT_3'$ для произвольного натурального числа n. Алгоритм выявления таких циклов вытекает из определения линейной дуги (см. раздел 3.2.1). Он не требует образования каких-либо циклов, но требует построения всех сильно связных компонент Dғрафа.

Непустые части сильно связных компонент, остающиеся за вычетом линейных дуг, определяют способные размножаться в маршрутах циклы и названы поэтому итерантами. Парность дуг Dғрафа обуславливает сцепленность итерантов, определяемую в разделе 3.2.2. Классы отношения сцепленности и характеризуют выделяемые в Dғрафе части.

В разделе 3.2.3 рассматриваются понятия форманта и развития, более общие по сравнению с определенными в главе 1. Обобщение сделано с целью обойтись более короткими маршрутами при отыскании циклов, обеспечивающих разрастание маршрутов. Здесь же определяются схемы — частный случай канонов.

В разделе 3.2.4 исследуется возможность переноса на случай Dғрафов техники расщепления сильно связных компонент, предложенной в разделе 3.1 в качестве основы построения регулярного выражения по конечному автомату. Выясняется, что попытка такого переноса наталкивается на значительные трудности, связанные с непредсказуемостью числа и характера вхождений парных дуг в циклы формантов. Устанавливаемая здесь верхняя оценка числа вхождений фиксированной пары дуг в схемы велика и влечет долгий перебор вариантов вхождений.

В разделе 3.2.5 дается метод разложения линий и схем, т.е. некоторых очень просто устроенных маршрутов, на элементы, определяющие производимые независимо друг от друга компоненты искомого КСвыражения. Формулируется алгоритм построения КСвыражения по Dғрафу.

Эквивалентное Dғрафу D КСвыражение, метод построения которого рассматривается в настоящей работе, обозначается через $Expr(D)$. Для построения выражения $Expr(D)$ строится выражение $\mathcal{E}(D)$ над алфавитом $E(D)$, определяющее язык $Sentences(D)$.

3.2.1. Лоскуты. Линии

Пусть

$$D = (V, \Sigma, \bar{\mathcal{P}}, \lambda, P_0, \bar{F})$$

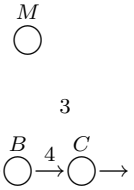
есть Dғраф, $\mathcal{P} \subseteq \bar{\mathcal{P}}$, $E \subseteq E(D)$, $I, F \subseteq V$. Тогда назовем пятерку

$$PD = (D, E, \mathcal{P}, I, F)$$

лоскутом D-*графа* D. Здесь I — множество *входных вершин лоскута* PD, F — множество его *заключительных вершин.*

Пусть $T \in E^*$ — маршрут Dғрафа D; пусть каждый нейтральный участок маршрута T является цепочкой из $\mathcal{L}_{\mathcal{P}}$. Тогда назовем T *маршрутом лоскута* PD. Если $pair(T) \in I \times F$, то назовем T *предложением лоскута* PD.

Пример 5. При определении маршрута лоскута нельзя ограничиться требованием $T \in E^*$. Действительно, рассмотрим Dғраф



с Dмножеством

$$\bar{\mathcal{P}} = \{(1,2), (3,4), (1,4), (3,2)\}$$

и его лоскут

$$PD = (D, \{1,2,3,4\}, \bar{\mathcal{P}} - \{(3,2)\}, \{A\}, \{C\}).$$

E^* содержит все маршруты Dғрафа D, число которых бесконечно; однако данный лоскут имеет только два предложения: 14 и 1234. Понятия, вводимые далее для

Dғрафов, нужно считать определенными и для их лоскутов. Например, можно говорить об итеранте лоскута и т.пр.

Заметим, что Dғраф, взятый целиком, можно считать его лоскутом

$$(D, E(D), \bar{\mathcal{P}}, \{P_0\}, \bar{F}).$$

Линии, маршруты в некотором смысле простейшего вида, служат частями выражения $\mathcal{E}(D)$. Кроме того, линии в графе, который является подграфом рассматриваемого Dғрафа и допускает в себе выделение подчиненных графов, предоставляют алгоритму выделения "отправные точки".

Следующее определение вводит характеристику маршрута, аналогичную понятию заряда маршрута магазинного автомата.

Пусть T — маршрут. Результат $\mu(T)$ примененения к нему Dөотображения назовем *зарядом* маршрута T.

Если $\mu(T) = \Lambda$, то будем называть маршрут T *нейтральным*.

Ясно, что заряд является цепочкой из $Right(\mathcal{P})^* Left(\mathcal{P})^*$ и, если не пуст, состоит из дуг, не имеющих пары в данном маршруте.

Пусть T — маршрут, $Closure(T) = \{T_1 T T_2 | T_1 T T_2$ — нейтральный маршрут; результат удаления из T_1 или T_2 некоторого непустого участка либо не является маршрутом, либо не нейтрален $\}$. Пусть $\hat{p} = \{(\pi_1, \pi_2) \in \mathcal{P} | \exists T' \in Closure(T)$ (π_1 и π_2 образуют гнездо в T')$\}$. Тогда будем говорить, что T есть *маршрут в* $\hat{p}$. Будем использовать для $\hat{p}$ обозначение $\mathcal{P}(T)$.

Пусть P и Q — вершины, w — заряд, $\hat{p} \subseteq \mathcal{P}$. Обозначим через

$$Computations(\hat{p}, P, Q, w)$$

множество всех маршрутов в $\hat{p}$ с парой концов (P, Q) и зарядом w. Если T — маршрут с парой концов (P, Q) и зарядом w, то будем считать запись "$Computations(\hat{p}, T)$" синонимом данного обозначения.

Таким же образом будем сокращать и некоторые далее определяемые обозначения, например, $Lines(\hat{p}, P, Q, w)$.

Элемент множества $Lines(\hat{p}, P, Q, w) = \{T \in Computations(\hat{p}, P, Q, w) | reduction(T) = T\}$ будем называть $(\hat{p}, P, Q, w)$-*линией* (или, просто, *линией*, если $\hat{p}$,P,Q,w либо подразумеваются, либо не существенны в данный момент).

Как видим, линия не содержит вставок, другими словами, не является производным маршрутом.

3.2.2. Итеранты и их сцепленность

Определим итеранты — связные компоненты Dғрафа, которые наряду с линиями играют ключевую роль в рассматриваемом построении. Для этого введем несколько вспомогательных понятий.

Далее нетривиальная, т.е. имеющая дуги, сильно связная компонента называется кратко сильно связной компонентой.

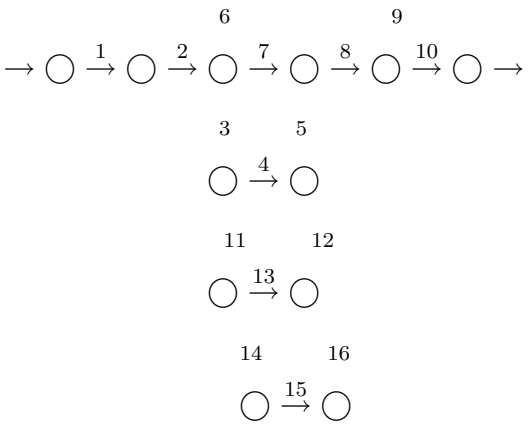
Пусть D — некоторый Dғраф. Дадим рекурсивное определение понятия *линейная* (*в* D) *дуга*:

1) дуга, которая не является дугой никакой сильно связной компоненты Dграфа D, называется линейной в нем;

2) пусть дуга π такова, что каждая парная ей дуга удовлетворяет пункту 1 данного определения; тогда π называется линейной в D;

3) пусть дуга π линейна в графе $descendant(D)$, получаемом из D удалением его дуг, удовлетворяющих пунктам $1, 2$ данного определения; тогда π считается линейной и в D.

Пример 6. В Dграфе D



с Dмножеством

$$\{(1,10),(2,3),(4,5),(4,11),(12,5),(7,8),(6,9),(13,14),(15,16)\}$$

дуги 1,2,7,8 и 10 линейны согласно первому пункту последнего определения, дуга 3 удовлетворяет второму пункту. Линейность дуг 4,5,11 и 12 обнаруживается в результате удаления дуг 1,2,3,7,8,10. В графе

$$descendant(descendant(D))$$

первому пункту определения линейной дуги удовлетворяет дуга 13, второму — 14, третьему — 15 и 16.

Из определения видно, что линейная дуга Dграфа участвует в линии, его собственной или некоторого "потомка", чем и объясняется выбор термина. Следующая лемма означает, что линейная дуга никогда не участвует в нейтральных циклах и циклах, образующих гнезда.

Лемма 18. Если дуга входит в нейтральный цикл или в какойнибудь из двух парных друг другу циклов, то она не является линейной.

Доказательство. Пусть π — линейная дуга в Dграфе D. Тогда существуют целое $k \geq 0$ и последовательность графов D_0,...,D_k, такие, что $D_0 = D$, $D_i = descendant(D_{i-1})$ для $0 < i \leq k$ и дуга π удовлетворяет в D_k пункту 1 или 2 определения линейной дуги. Докажем лемму индукцией по k.

Заметим предварительно, что если некоторая дуга π_1 удовлетворяет в некотором графе пунктам 1,2 определения линейной дуги, то она сама или любая парная ей дуга π_2 не принадлежит никакой сильно связной компоненте этого графа. Согласно определениям гнезда и нейтрального цикла каждая дуга одного из двух парных циклов имеет парную в нем самом или в парном ему цикле, а каждая дуга

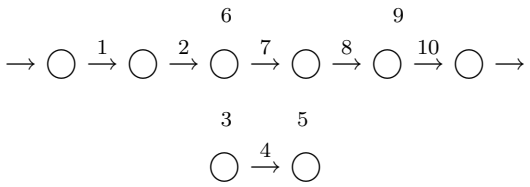
нейтрального цикла имеет парную в нем самом. Но это означает, что элемент D-множества, образованный дугами π_1 и π_2, не может участвовать в нейтральных и парных циклах рассматриваемого графа.

Из установленного следует, что маршруты графа D_i, $0 \le i \le k$, не содержат нейтральных или парных циклов, в которых участвовали бы дуги, удовлетворяющие в нем пунктам 1,2 определения линейной дуги. Таким образом, утверждение справедливо при $k = 0$.

Пусть $k > 0$. Допустим, что дуги графа D_k, удовлетворяющие в нем пунктам 1,2 определения линейной дуги, входят в нейтральные или парные циклы объемлющего графа. Тогда согласно установленному выше в этих циклах обязаны участвовать дуги, отсутствующие в D_k. Но участие этих дуг в нейтральных или парных циклах противоречит предположению индукции □

Пусть сильно связная компонента G Dғрафа D имеет дуги, которые не являются линейными в D. Пусть ее подграф C определяется множеством всех таких дуг и их вершинами. Тогда назовем C *итерантом*.

Заметим, что сильно связная компонента может содержать как линейные дуги, так и дуги, не являющиеся линейными. Так, в Dғрафе



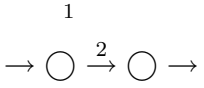
с Dмножеством

$$\{(1,4),(2,3),(5,10),(6,9),(7,8)\}$$

дуги $3,4,5$ линейны, а дуга 6 образует итерант. Второй итерант данного Dғрафа состоит из дуги 9.

Может случиться, что все дуги сильно связной компоненты линейны, как в следующем примере. Таким образом, сильно связной компоненте Dғрафа отвечает не больше одного итеранта.

Пример 7. Следующий Dғраф с Dмножеством $\{(1,2)\}$ не имеет итерантов, хотя у него есть нетривиальная сильно связная компонента.



Лемма 19. Итерант сильно связен.

Доказательство. Противное означало бы, что среди дуг итеранта есть линейные в D дуги, вопреки определению итеранта □

Следующий факт подготавливает, по существу, осознание возможности важной взаимозависимости различных итерантов.

Лемма 20. Любая дуга итеранта парна некоторой дуге некоторого итеранта.

Доказательство. Предположим, что некоторая дуга π некоторого итеранта образует пары только с дугами, не входящими в итеранты, т.е. линейными. Но тогда и дуга π линейна, вопреки определению итеранта □

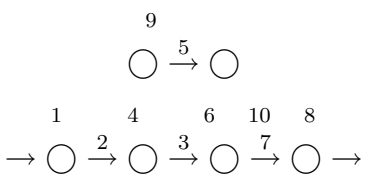
Эта лемма означает, что дуги нейтральных и парных циклов являются дугами некоторых итерантов. Другими словами, верно утверждение, обратное лемме 18.

Пусть $(\pi_1, \pi_2) \in \mathcal{P}$. Пусть для $i = 1, 2$ дуга π_i является дугой итеранта C_i (допускается равенство $C_1 = C_2$). Тогда будем говорить, что итеранты $_1$ и $_2$ *сцеплены*. Итерант C, сцепленный с C_1 (или C_2), считаем сцепленным с C_2 (с C_1 соответственно).

Таким образом, на множестве итерантов Dғрафа определяется рефлексивное, симметричное и транзитивное отношение сцепленности.

Пример 8. Пусть для Dғрафа, представленного следующим рисунком,

$$\mathcal{P} = \{(1,4),(2,3),(5,6),(5,10),(1,7),(9,8)\}.$$



Тогда его три сильно связные компоненты сцеплены друг с другом.

Здесь дуги $1, 5$ парны соответственно линейным дугам $7, 10$. Однако они являются дугами итерантов, так как парны соответственно дугам $4, 6$ сильно связной компоненты, не имеющей линейных в данном Dғрафе дуг. Правда маршрутов, в которых дуги 1 и 4 образуют гнезда, не существует. В самом деле, единственный путь из $end(1)$ в $beg(4)$ состоит из дуги 2, т.е. не является нейтральным маршрутом.

3.2.3. Расширение понятия форманта

Рассматриваемое расширение позволит при обработке Dғрафа обходиться более короткими маршрутами на итерантах. В число формантов будут включены и некоторые нейтральные маршруты. Для построения ненейтрального форманта, достижимого из некоторой фиксированной вершины итеранта, требуется "пройти", вообще говоря, более короткий маршрут, чем в случае, когда нейтральность форманта была обязательна.

3.2.3.1. Вставки и схемы. В определении вставки фиксируется наблюдение о том, что для разрастания маршрутов можно использовать и некоторые ненейтральные циклы.

Схемы определяются в терминах вставок. Они родственны канонам весьма частного вида.

Пусть T — цикл, $\mu(T) = T_1T_2$, $T_1 \in Right(\mathcal{P})^*$, $T_2 \in Left(\mathcal{P})^*$, $\mu(T_2T_1) = \Lambda$. Тогда назовем цикл T *Rвставкой* .

Заметим, что $T^k, k \geq 1$, является циклическим маршрутом и $\mu(T^k) = \mu(T)$. Является маршрутом и путь T_3T_4, если T_3TT_4 — маршрут, в котором T есть Rвставка. Таким образом, замена Rвставки любой ее степенью (в том числе и нулевой) оставляет маршрут маршрутом. Это оправдывает употребление в термине слова "вставка".

Указанный класс маршрутов напоминает о регулярных языках, чем и объясняется приставка "R".

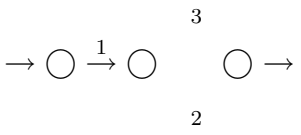
Нейтральный цикл есть частный случай Rвставки.

Пусть тройка (T_1, T_2, T_3) такова, что $T_1T_2T_3$ — маршрут, T_1 и T_3 — циклы, $\mu(T_1) = T_{11}T_{12}T_{13}$, $\mu(T_2) = T_{11}T_{33}$, $\mu(T_3) = T_{31}T_{32}T_{33}$, T_{12} и T_{32} не пусты, $\mu(T_{12}T_{32}) = \Lambda$, $\mu(T_{13}T_{11}) = \mu(T_{33}T_{31}) = \Lambda$. Тогда будем говорить, что данная тройка *указывает повторяемые* (точнее, *согласованно повторяемые*) *циклы* T_1 и T_3. Маршрут $T_1T_2T_3$ назовем S*-вставкой.*

Заметим, что $T_1^kT_2T_3^k, k \geq 0$, является маршрутом с зарядом $T_{11}T_{33}$. Парные циклы — частный случай повторяемых циклов.

Понятие Sвставки вызывает ассоциации с понятием самовставляющегося символа грамматики, отсюда приставка "S".

Пример 9. Dграф

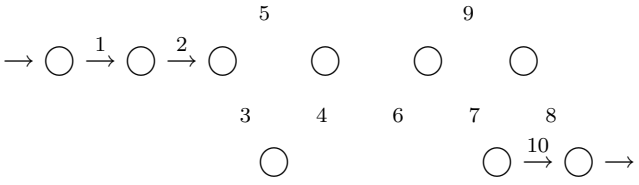


с Dмножеством

$$\{(1,2),(3,2)\}$$

определяет Rвставку 23, которая удовлетворяет определению элементарной схемы (см. ниже). Элементарной схемой является и предложение 1232.

Пример 10. Dграф



с Dмножеством

$$\{(1,10),(2,3),(4,8),(5,3),(6,7),(9,7)\}$$

определяет Sвставку

$$T = 34536789,$$

которая является участком элементарной схемыпредложения

$$(1,2,T,7,10).$$

Здесь повторяемы ненейтральные циклы 345 и 789.

Примеры показывают, что сформулированные в настоящей работе определения R- и Sвставок позволяют обнаружить явления итерации и согласованной итерации подцепочек в цепочках заданного Dграфом языка, минимально "прокручиваясь" по сильно связным компонентам.

Маршрут, который является Rвставкой или Sвставкой, будем называть *вставкой.*

Формант — это вставка, любой собственный участок которой не является вставкой.

Схема — маршрут, все вставки которого являются формантами.

Собственный участок вставки T, входящей в некоторую схему, не может сам быть вставкой: иначе T не была бы формантом. Однако различные вставки одной схемы могут иметь общий участок.

Пусть в схеме T любые две вставки имеют непустой общий участок. Тогда назовем T *элементарной схемой.*

Итак, понятие элементарной схемы расширено по сравнению с [Вылиток 98] не только за счет более емкого понятия вставки, но и за счет разрешения нескольких определенным образом "пересекающихся"вставок.

Действительно, чтобы попасть из одной заданной вершины итеранта в другую заданную вершину, используя некоторую вставку, но не используя одновременно вставок, не имеющих непустых общих участков, может потребоваться обойти все циклы пучка циклов, похожего на демонстрируемый следующим примером маршрута. Поэтому кажется естественным считать элементарной схему

$$T_1T_2T_3T_4T_5T_6T_7,$$

где T_i — нейтральный маршрут для $1 \leq i \leq 7$, начальные вершины маршрутов T_1, T_2, T_3 и T_4 попарно различны, $T_1T_2T_3T_4$, $T_2T_3T_4T_5$, $T_3T_4T_5T_6$ и $T_4T_5T_6T_7$ являются Rвставками.

Лемма 21. Пусть T — нейтральный участок элементарной схемы. Тогда справедливы неравенства

$$width(T) \leq min(|V|+1, |\mathcal{P}|),$$
$$depth(T) \leq min(|V|^2, |\mathcal{P}|+1).$$

Доказательство. Первое из неравенств доказывается аналогично лемме 2 из [Вылиток 98], а второе — аналогично лемме 3, оттуда же □

Лемма 22. Пусть T' — нейтральный участок элементарной схемы T, $m = |V|$, $p = |\mathcal{P}|$, $n = g_{min(m+1,p),min(m^2,p+1)}$. Тогда

$$|T'| \leq n,\ |T| \leq n(|\mu(T)|+1).$$

Доказательство. Доказательство первого неравенства аналогично доказательству леммы 4 из [Вылиток 98]. Второе неравенство следует из представления элементарной схемы T последовательностью

$$T_0\pi_1T_1...\pi_{|\mu(T)|}T_{|\mu(T)|},$$

где $\mu(T) = \pi_1...\pi_{|\mu(T)|}$, T_i — нейтральный участок для $0 \leq i \leq |\mu(T)|$ □

Из леммы 22 следует

Теорема 9. Множество элементарных схем, имеющих одни и те же заряд и пару концов, конечно □

3.2.3.2. Отношение развития маршрутов. Развитие (по другому можно сказать "разрастание") сводится к последовательности замен так называемых атомов формантами. Как видно из следующих определений, атом всегда присутствует во вставке, в частности, в форманте, и может встречаться в маршрутах, которые не содержат вставок.

Пусть T — это Rвставка. Тогда назовем *R-атомом* (вставки T) пустой маршрут с вершиной $beg(T)$.

Пусть тройка (T_1, T_2, T_3) указывает согласованно повторяемые циклы T_1 и T_3. Тогда назовем T_2 *S-атомом* (вставки $T_1T_2T_3$).

Маршрут, который является Rатомом или Sатомом, будем называть *атомом.*

Заметим, что атом является элементарной схемой, не содержащей вставок.

Определим на множестве маршрутов Dграфа D бинарное отношение $\Uparrow_D$ *непосредственного развития* следующим образом.

$$(T_1T_3T_5, T_1T_2T_3T_4T_5) \in \Uparrow_D$$

с историей развития

$$(T_1, T_2, T_3, T_4, T_5)$$

тогда и только тогда, когда $T_2T_3T_4$ есть формант с атомом T_3. Отношение $\Uparrow_D^*$ назовем *отношением развития*.

Из определений можно вывести следующее утверждение.

Лемма 23. Пусть T — маршрут Dграфа D. Тогда существуют такие число k и последовательность маршрутов $T_0, ..., T_k$, что T_0 не содержит вставок, $T_k = T$ и для $0 \le i < k$ справедливы соотношения $pair(T_i) = pair(T)$, $\mu(T_i) = \mu(T)$, $(T_i, T_{i+1}) \in \Uparrow_D$ □

Заметим, что обсуждаемая в лемме 23 последовательность определена неоднозначно. Любую из таких последовательностей будем называть *родословной маршрута* T (*с предком* T_0). Ясно, что

$$T_0 \in Lines(\mathcal{P}, beg(T), end(T), \mu(T)).$$

Любое множество $Lines(\mathcal{P}, P, Q, w)$, где P и Q — вершины, w — заряд, вполне аналогично множеству $Frame(M)$, определенному в [Вылиток 98] для магазинного автомата M. Действительно, множество $Lines(\mathcal{P}, P, Q, w)$ есть множество "простейших" предков маршрутов из $Computations(\mathcal{P}, P, Q, w)$.

Теорема 2 из [Вылиток 98] представляет собой частный случай следующей теоремы, вытекающей непосредственно из определения родословной.

Теорема 10. Пусть $T \in Computations(\mathcal{P}, P, Q, w)$. Тогда существуют такие целое $k \ge 0$ и маршрут $T' \in Lines(\mathcal{P}, P, Q, w)$, что $(T', T) \in \Uparrow_D^k$ □

Схемы и их подмножества, именно линии и элементарные схемы, играют важную роль в рассматриваемом алгоритме построения КСвыражения по Dграфу. Эффективность построения схем влияет решающим, пожалуй, образом на эффективность всего алгоритма.

3.2.4. Некоторые наблюдения об устройстве схем

3.2.4.1. О повторяемости в схемах. Графы, подчиненные Dграфу, естественно определять как части его итерантов, позволяющие строить маршруты, которые не используют некоторой пары дуг итерантов. (Заметим, что запрещение использовать некоторую пару из Dмножества не всегда сопряжено с изъятием из графа дуг этой пары: ведь они могут участвовать в других парах данного Dмножества.)

Законно предположить, что выделение подграфов может быть более или менее удачным и что удача зависит от выбора исключаемой пары. В связи с этим важно знать характер повторяемости в схемах одной и той же пары дуг итерантов.

Возможны следующие два взаимоисключающих случая:

1) класс сцепленности содержит только один итерант;

2) класс содержит больше одного итеранта.

В первом случае все относящиеся к итеранту пары дуг образуют гнезда в некоторых циклах и в этом смысле равноправны.

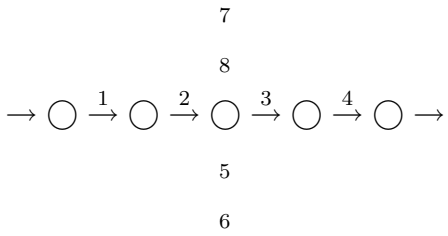
Во втором случае относящиеся к итерантам класса пары дуг можно разделить на два сорта:

1) участвующие в маршрутах только некоторого одного итеранта (таким образом, обе дуги пары являются дугами этого итеранта);

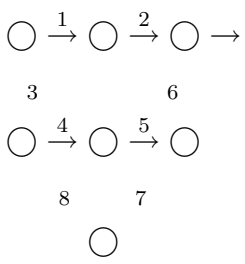
2) образованные дугами двух различных итерантов. Для класса неединичной мощности пары первого из сортов могут отсутствовать, но пары второго обязательны (иначе итеранты класса не были бы сцеплены).

Рассмотрим несколько примеров Dғрафов и их схем.

Пример 12. Dғрафы



и



с Dмножествами

$$\{(1,4),(2,3),(5,6),(7,8)\}$$

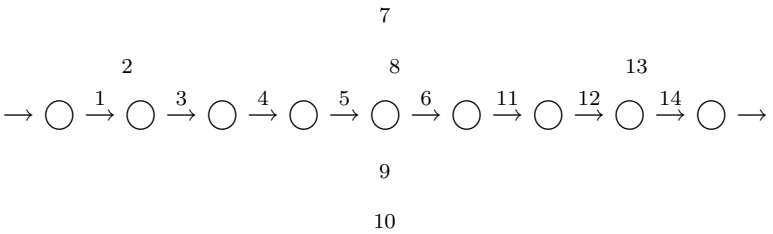
и

$$\{(1,3),(6,2),(6,3),(4,5),(7,8)\}$$

соответственно имеют по одному итеранту. Первый граф определяет среди прочих схемы 235236 и 1234712348, второй — 134562 (это предложение), 3456 и 4578 (это Rвставки, ненейтральная и нейтральная соответственно.)

Любопытная особенность данного примера — неоднократное использование некоторых пар дуг не только в различных схемах, но и в одной и той же схеме.

Пример 13. Dграф



с Dмножеством

$$\{(1,14),(2,7),(3,4),(5,6),(8,13),(9,10),(11,12)\}$$

определяет как Rвставки, так и Sвставки. Так, (5,6,9,10) — нейтральная R-вставка, а (2,3,4,5,6,7) и (8,5,6,11,12,13) — нейтральные Sвставки.

Сильно связные компоненты данного Dграфа являются его итерантами. Все они сцеплены. Пара (5,6) используется в каждой вставке, хотя относится к одному итеранту.

Далее будем обозначать через $\mathcal{P}(C)$ множество всех пар из $\mathcal{P}$, которые участвуют во вставках, концы которых являются вершинами итерантов данного класса.

Заметим, что в случае $|C| > 1$ среди дуг, которые входят в эти пары, могут быть линейные. Такова дуга 6 в примере 4.2.

Соответственно классу сцепленности C определим множество его "расщепителей"как следующую часть Dмножества:

$$Splitters(C) = \mathcal{P}(C) - \{(\pi_1, \pi_2) | \text{ хотя бы одна из дуг } \pi_1, \pi_2 \text{ линейна}\}.$$

Дуги расщепителя образуют гнездо в некоторой Rвставке или Sвставке. В последнем случае либо они образуют гнездо в одном из повторяемых циклов, либо левая дуга входит в левый из повторяемых циклов, а правая — в соответствующий правый. Иное противоречило бы "скобочной"интерпретации элемента Dмножества.

3.2.4.2. Оценка числа повторов расщепителей в некоторых схемах

Для элементарной схемы T и пары $(\pi_1, \pi_2) \in \mathcal{P}$ обозначим через $\nu(T, \pi_1, \pi_2)$ число гнезд в T, которые имеют вид (π_1, T', π_2).

Пусть для гнезд (π_1, T, π_2) и (π_1', T', π_2') справедливо равенство $(\pi_1, \pi_2) = (\pi_1', \pi_2')$. Тогда назовем эти гнезда *одноименными*.

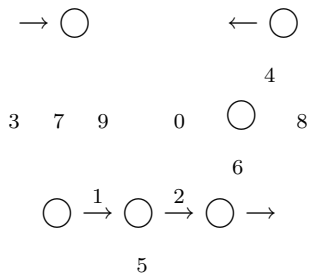
Рассмотренные выше примеры показывают, что для класса единичной мощности дуги его расщепителя могут образовывать несколько (см. ниже некоторые численные оценки) гнезд в Rвставкеєхеме.

Следующий пример показывает, что в Sвставкеєхеме также может быть много одноименных гнезд. Согласно следующей лемме возможность использования одного такого гнезда внутри другого довольно жестко ограничена.

Лемма 24. Пусть C — класс сцепленности, $(\pi_1, \pi_2) \in Splitters(C)$ и вставка T в $\mathcal{P}(C)$ является схемой. Тогда глубина скобочной системы, получаемой из T удалением каждой пары дуг, отличной от (π_1, π_2), не превосходит двух.

Доказательство. В противном случае вставка T не была бы схемой □

Пример 14. Dграф



с Dмножеством

$$\{(1,2),(3,4),(5,6),(7,8),(9,0)\}$$

совпадает со своей сильно связной компонентой C. $Splitters(C) = \mathcal{P}(C) = \mathcal{P}$. В предложенииєхеме 71931251264028 дуги расщепителя (1,2) образуют три гнезда. Указанная леммой скобочная система есть 112122. Ее глубина равна двум.

При оценке числа одноименных гнезд в элементарной схеме важно следующее утверждение.

Лемма 25. Пусть T — элементарная схема без Sвставок (таковы, в частности, линия и формант, не являющийся Sвставкой), T_1 и T_2 — ее одноименные гнезда. Тогда: 1) ни одно из гнезд не является участком другого; 2) для любого нейтрального маршрута T_3 маршруты $T_1T_3T_2$ и $T_2T_3T_1$ не являются участками схемы T.

Доказательство. Действительно, в первом случае схема содержала бы S-вставку, во втором собственный участок схемы (например, T_1T_3) являлся бы R-вставкой □

Лемма 26. Пусть T — элементарная схема без S-вставок, $\nu = max\{\nu(T, \pi_1, \pi_2) | (\pi_1, \pi_2) \in \mathcal{P}\}$, $p = parti(T)$, $w = width(T)$, $d = depth(T)$. Тогда

$$\nu \leq \begin{cases} p \cdot w^{d-2}, d > 1, \\ 1, d = 1. \end{cases}$$

Доказательство. Представим T в виде $T = T_1...T_p$, где T_i имеет протяжение 1. Заметим, что скобочная система протяжения 1 может быть представлена ориентированным деревом, корень которого отвечает паре внешних скобок. Из вершины, отвечающей некоторым парным скобкам, исходит столько дуг, каково протяжение заключенной в этих скобках подсистемы.

Следовательно, маршрут T_i, $i = 1, ..., p$, можно изобразить поддеревом w-ичного дерева высоты d, имеющего w^{d-1} листьев.

Вследствие леммы 24 о взаимном расположении одноименных гнезд в элементарной схеме величина $\nu(T_i, \pi_1, \pi_2)$ оценивается числом w^{d-2} при $d > 1$ и единицей при $d = 1$ для любой пары $(\pi_1, \pi_2) \in \mathcal{P}$. Действительно, среди конечных вершин дуг с общей начальной вершиной одна может отвечать паре (π_1, π_2), при условии что путь к ней от корня не содержит других вершин, отвечающих той же паре.

В случае $d > 1$ лемма следует из неравенства $\nu(T, \pi_1, \pi_2) \leq p \cdot max\{\nu(T_i, \pi_1, \pi_2) | 1 \leq i \leq p\}$. В случае $d = 1$ справедливо равенство $\nu(T_i, \pi_1, \pi_2) = 1$ в силу леммы о взаиморасположении одноименных гнезд □

Из леммы 26 и оценок ширины и глубины элементарной схемы следует, в частности, оценка

$$\nu(T, \pi_1, \pi_2) \leq |\mathcal{P}|^{|\mathcal{P}|}.$$

3.2.4.3. Секущие и их применение для классификации маршрутов, содержащих вершины итерантов. Ясно, что маршрут из вершины итеранта в вершину итеранта может быть сокращен удалением R-вставок и повторяемых циклов до некоторой линии. Некоторые из линий, содержащих вершины итерантов, назовем секущими (см. определение ниже) и применим для описания маршрутов, не являющихся линиями.

Пусть маршрут $T = T_0T_1T_2$ является линией, $beg(T_1)$ и $end(T_1)$ вершины итерантов некоторого класса C, T_0 и T_2 не содержат дуг итерантов того же класса. Тогда назовем T_1 *секущей для класса* C (*в маршруте* T).

Отметим, что если T является предложением и заряд $\mu(T_1)$ непуст, то образующие его дуги парны в данном предложении некоторым линейным дугам.

Заметим, что может быть несколько секущих с одинаковым зарядом и парой концов, и обозначим через $Across(C, P, Q, w)$ множество всех секущих для класса C, которые имеют пару концов (P, Q) и заряд w.

Пусть

$$T \in Across(C, P, Q, w),$$
$$(\pi_1, \pi_2) \in Splitters(C),$$
$$\hat{p} = \mathcal{P}(C) - \{(\pi_1, \pi_2)\},$$
$$pair(\pi_i) = (P_i, Q_i) \text{ для } i = 1, 2.$$

Тогда в случае $|C| > 1$ обозначим через $Code(T, \pi_1, \pi_2)$ объединение множеств

$$Code_1(T, \pi_1, \pi_2) = \{(T_1, T_2, T_3) | T_1 \in Lines(\hat{p}, P, P_1, w_1),$$
$$T_2 \in Lines(\hat{p}, Q_1, P_2, \Lambda),\ T_3 \in Lines(\hat{p}, Q_2, Q, w_2);\ \mu(w_1 w_2) = w\}$$

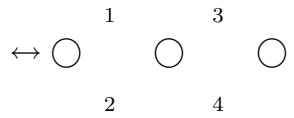
и

$$Code_2(T, \pi_1, \pi_2) = \{(T_1, T_2, T_3, T_4, T_5) |$$
$$T_1 \in Lines(\hat{p}, P, P_1, w_1),\ T_2 \in Lines(\hat{p}, Q_1, P_1, w_3),$$
$$T_3 \in Lines(\hat{p}, Q_1, P_2, \Lambda),\ T_4 \in Lines(\hat{p}, Q_2, P_2, w_4),$$
$$T_5 \in Lines(\hat{p}, Q_2, Q, w_2);\ \mu(w_1 w_2) = w,\ \mu(w_3 w_4) = \Lambda\}.$$

Пусть C — класс сцепленности, $|C| = 1$. Пусть $T \in Across(C, P, Q, w)$, $(\pi, \pi') \in Splitters(C)$, $\hat{p} = \mathcal{P}(C) - \{(\pi, \pi')\}$, $(\pi_1, \pi_2) \in \{(\pi, \pi'), (\pi', \pi)\}$, $pair(\pi_i) = (P_i, Q_i)$ для $i = 1, 2$, $m = max\{\nu(T', \pi, \pi') |\ T'$ — формант, $beg(T')$ — вершина итеранта из $C\ \}$. Тогда обозначим через $Code(T, \pi, \pi')$ объединение $2m^2$ множеств $Code_{k,l}(T, \pi_1, \pi_2)$, $1 \le k,\ l \le m$:
$Code_{k,l}(T, \pi_1, \pi_2) = \{(T_{1,1}, ..., T_{1,2k+1}, T_{2,1}, ..., T_{2,2l+1} |$
$T_{1,1} \in Lines(\hat{p}, P, P_1, w_{1,1})$;
$T_{1,j} \in Lines(\hat{p}, Q_1, P_2, w_{1,j}), j = 2, ..., 2k$;
$T_{1,j+1} \in Lines(\hat{p}, Q_2, P_1, w_{1,j+1}), j = 2, ..., 2k$ при $k > 1$;
$T_{1,2k+1} \in Lines(\hat{p}, Q_2, P, w_{1,2k+1})$;
$T_{2,1} \in Lines(\hat{p}, P, P_1, w_{2,1})$;
$T_{2,j} \in Lines(\hat{p}, Q_1, P_2, w_{2,j}), j = 2, ..., 2l$;
$T_{2,j+1} \in Lines(\hat{p}, Q_2, P_1, w_{2,j+1}), j = 2, ..., 2l$ при $l > 1$;
$T_{2,2l+1} \in Lines(\hat{p}, Q_2, Q, w_{2,2l+1})$;
$T' = T_{1,1}\pi_1 T_{1,2}\pi_2 ... \pi_1 T_{1,2k}\pi_2 T_{1,2k+1}$ — Rвставка;
$\mu(T'T'') = \mu(T'') = w$ для $T'' = T_{2,1}\pi_1 T_{2,2}\pi_2 ... \pi_1 T_{2,2l}\pi_2 T_{2,2l+1}\}$.

Следующий пример показывает, что если ни одна из дуг расщепителя (π, π') не инцидентна какой-либо вершине секущей T, то маршрут $T'T''$ может и не быть схемой. Однако "перемычки" T_{ij} между ближайшими друг к другу вхождениями дуг расщепителя являются линиями (не обязательно нейтральными).

Пример 15. Dграф



с Dмножеством

$$\mathcal{P} = \{(1, 2), (3, 4)\}$$

совпадает со своим единственным итерантом. Пустое предложение T является секущей. Указываемый элементом множества

$$Code(T, 3, 4) = \{(1, end(3), 2)\}$$

маршрут $(1, 3, 4, 2)$ есть Rвставка, но не формант, так как содержит собственную Rвставку $(3, 4)$.

3.2.5. Специальные разложения маршрутов

В терминах сцепленных итерантов определим способ разложения маршрутов на последовательные участки, который поможет выделять подчиненные Dграфы при построении по Dграфу КСвыражения.

3.2.5.1. Разделение Dграфа, обусловленное отношением сцепленности. Пусть $T \in Computations(\bar{p}, P, Q, w)$. В простейшем случае среди вершин маршрута T нет вершин итерантов. Тогда $T \in Lines(\bar{p}, P, Q, w)$.

Если какая-либо из вершин маршрута T является вершиной итеранта, то его можно представить последовательностью $T_0T_1T_2$, где T_0 не содержит дуг итерантов, пара $pair(T_1)$ образована вершинами сцепленных итерантов, T_2 не содержит дуг итерантов, сцепленных с представленными в T_1 итерантами.

Тройку (T_0, T_1, T_2) обозначим через $fi(\bar{p}, T)$. Из определений итеранта и сцепленности итерантов следует единственность такого разложения маршрута T.

Пусть C — класс, содержащий итеранты с вершинами $beg(T_1)$, $end(T_1)$. Участку T_2 сопоставим лоскут

$$Rest(D, T_2) = (D, E(D), \mathcal{P} - \{(\pi_1, \pi_2) \in \mathcal{P}(C) | \text{дуга } \pi_1 \text{ или } \pi_2 \text{ не является линейной}\}, \{beg(T_2)\}, \{end(T_2)\}).$$

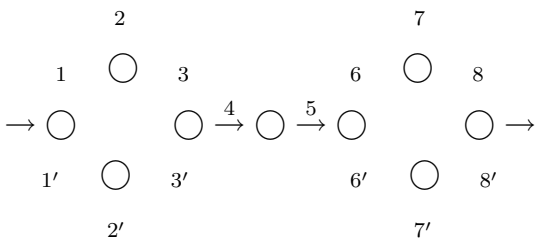
Заметим, что в случае пустоты участка T_2 лоскут $Rest(D, T_2)$ не имеет непустых предложений. Действительно, при $beg(T_2) = end(T_2) = P'$ непустой нейтральный маршрут с парой концов (P', P') является Rвставкой. Значит, участвующие в нем пары Dмножества входят в $\mathcal{P}(C)$ и состоят из дуг, которые не являются линейными. Следовательно, рассматриваемый маршрут не может быть предложением лоскута $Rest(D, T_2)$. Таким образом,

$$Sentences(Rest(D, T_2)) = \{beg(T_2)\}$$

при пустом T_2 и язык $L(Rest(D, T_2))$ всегда непуст.

Dмножество лоскута $Rest(D, T_2)$ нельзя определять как $\mathcal{P} - \mathcal{P}(C)$. Следующий пример показывает, что $\mathcal{P}(C)$ может вовлекать линейные дуги Dграфа D, участвующие во вставках, определяемых некоторым классом $C' \neq C$.

Пример 16. Рассмотрим Dграф



с Dмножеством

$$\mathcal{P} = \{(1, 8), (2, 7), (3, 6), (4, 5), (1', 8'), (2', 7'), (3', 6')\}.$$

Его итеранты образуют два класса сцепленности, каждый из которых определяет Sвставки, использующие одну и ту же пару линейных дуг $(4, 5)$.

3.2.5.2. Разложение секущей и основной алгоритм. Ограничимся далее нейтральными атомами и вставками.

Пусть $T' \in Across(C, P, Q, w)$. Пусть T — нейтральный участок секущей T', который нельзя увеличить (за счет соседних участков), не нарушая его нейтральности. Назовем такой участок *звеном*. Обозначим через $Segmentations(T)$ множество всех последовательностей вида $T_0T_1'T_1...T_k'T_k$, где для $1 \leq i \leq k$ T_i' есть

атом, для $0 \le j \le k$ T_j не содержит атомов или пуст и T_j непуст, если $T'_j = T'_{j+1}$ есть Rатом.

Последнее требование обеспечивает конечность множества $Segmentations(T)$.

Для каждого звена T построим с помощью множества $Segmentations(T)$ совокупность всех "потомков" звена, удовлетворяющих определению (1,2)канона, и расставим в каждом потомке именованные скобки по принципу, изложенному в параграфе 1.3 при определении так называемой схемы фразы. Обозначим результат расстановки скобок в t через $scheme(t)$. Пусть

$$Schemes(T) = \{scheme(t) | t \text{ является (1,2)каноном; } (T,t) \in \Uparrow^*\}.$$

Пусть T — секущая, $T_1, ..., T_l$ — все ее звенья. Обозначим через $\sigma(T)$ множество $\{t_0\sigma_1 t_1 ... \sigma_l t_l |\ t_0 T_1 t_1 ... T_l t_l = T;\ \sigma_i \in Schemes(T_i)\}$.

Заметим, что участки t_j, $0 \le j \le l$, секущей T определяются единственным образом. Обозначение $\sigma(T)$ имеет смысл и при отсутствии в T звеньев, то есть при $l = 0$. Тогда $t_0 = T$, $\sigma(T) = \{T\}$.

Алгоритм построения КСвыражения $\mathcal{E}(D)$ обозначим так же, как искомое выражение. Можно считать его функцией, отображающей Dграфы в КСвыражения. Значение функции вычисляется рекурсивным образом.

Алгоритм 7. Преобразование Dграфа в КСвыражение, управляемое фактормножеством на множестве итерантов по отношению сцепленности.

Вход. Dграф $D = (V, \Sigma, \mathcal{P}, \lambda, P_0, F)$.

Выход. КСвыражение $\mathcal{E}(D)$.

Вспомогательные обозначения.

$\Phi(D) = \{fi(T) |\ T \in Lines(\mathcal{P}, P_0, Q, \Lambda);\ Q \in F\}$;

$Straight(D) = \{T \in Lines(\mathcal{P}, P_0, Q, \Lambda) |\ Q \in F;\ T$ — не имеет вершин итерантов $\}$.

Метод. Положить $\mathcal{E}(D)$ равным

$$\sum_{T \in Straight(D)} T + \sum_{(\hat{T_0}, \hat{T_1}, \hat{T_2}) \in \Phi(D)} [\hat{T_0}(\sum_{T \in Across(\hat{T_1})} \sigma(T)) \mathcal{E}(Rest(D, \hat{T_2}))].$$

Теорема 11. Пусть D — Dграф или лоскут. Тогда $L(\mathcal{E}(D)) = Sentences(D)$.

Доказательство. Индукцией по числу классов сцепленности. Если таковые отсутствуют, то

$$Sentences(D) = Straight(D),$$

$$\mathcal{E}(D) = \sum_{T \in Sentences(D)} T;$$

таким образом, теорема в данном случае верна.

Пусть D имеет классы сцепленности. Тогда $\Phi(D) \neq \emptyset$. Заметим, что каждое предложение, не принадлежащее $Straight(D)$, можно представить в виде $\hat{T_0} T_1 T_2$, где для некоторых $\hat{T_1}$, $\hat{T_2}$ тройка $(\hat{T_0}, \hat{T_1}, \hat{T_2}) \in \Phi(D)$ и $(\hat{T_i}, T_i) \in \Uparrow^*_D$ для $i = 1, 2$.

Маршрут $\hat{T_0}$ способен развиваться разве только за счет своей конечной вершины, что можно игнорировать, так как эту вершину $end(\hat{T_0}) = beg(\hat{T_1})$ можно отнести и к $\hat{T_1}$. Следовательно, входящая в $\mathcal{E}(D)$ сумма по элементам множества $\Phi(D)$ задает все способные к развитию предложения, если подвыражение $\xi_1 \xi_2$,

где

$$\xi_1 = \sum_{T \in Across(\hat{T}_1)} \sigma(T)$$

и

$$\xi_2 = \mathcal{E}(Rest(D, \hat{T}_2)),$$

задает все маршруты с парой концов $(beg(\hat{T}_1), end(\hat{T}_2))$ и зарядом $\mu(\hat{T}_1\hat{T}_2)$. Согласно параграфу 1.3 ξ_1 задает множество $Computations(\mathcal{P}(C), \hat{T}_1)$. По предположению индукции ξ_2 задает множество $Sentences(Rest(D, \hat{T}_2))$, представляющее все маршруты с парой концов $pair(\hat{T}_2)$ и зарядом $\mu(\hat{T}_2)$. Следовательно, теорема верна
□

Глава 3

НЕКОТОРЫЕ АЛГОРИТМИЧЕСКИЕ ПРОБЛЕМЫ

Большая часть данной главы посвящена алгоритмическим проблемам, связанным с проблемой эквивалентности детерминированных магазинных автоматов.

Параграф 1 затрагивает сложную тему поиска эффективных алгоритмов, проверяющих регулярность детерминированного языка. В нем представлено достаточное условие регулярности детерминированного языка, проверяемое по ядру определяющего этот язык детерминированного Dғрафа или детерминированного магазинного автомата. Отметим, что этот результат был усилен в диссертации [Вылиток 98].

В параграфе 2 определяется понятие графа сопряжений (естественных сопоставлений путей, определяющих одну и ту же цепочку языка) двух Dғрафов. Доказывается неразрешимость алгоритмической проблемы построения графа сопряжений для графов детерминированных магазинных автоматов, наполняющих магазин в реальное время. Проблема формулируется следующим образом.

> Построить алгоритм, который по произвольной паре эквивалентных Dғрафов получает их граф сопряжений.

Так как граф сопряжений графов двух эквивалентных магазинных автоматов является квинтэссенцией всевозможных устройств, имитирующих одновременную работу автоматов над одной и той же цепочкой, установленный факт отвергает попытки применить технику имитации для проверки эквивалентности автоматов указанного класса. Отмечается, что эта техника неприменима для решения проблемы эквивалентности даже в классе детерминированных магазинных автоматов, действующих в реальное время.

Параграф 3 показывает полезность понятия ядра для получения математически изящных решений некоторых проблем и устанавливает алгоритмическую неразрешимость проблемы эквивалентности памятей детерминированного магазинного автомата, действующего в реальное время. Памятью названа пара

(состояние, непустая цепочка магазинных символов),

которая входит в состав некоторой конфигурации магазинного автомата. При этом второй элемент пары является, вообще говоря, лишь некоторой верхней частью содержимого магазина. Указываются подклассы детерминированных магазинных автоматов, в которых последняя проблема и, следовательно, проблема эквивалентности самих автоматов алгоритмически разрешима. Проверка эквивалентности автоматов указанного класса сводится к проверке эквивалентности их начальных памятей (начальная память определяется начальным состоянием и магазинной цепочкой из одного символа — маркера дна).

Кроме того, в параграфе 3 доказывается неразрешимость проблемы регулярности объединения двух непустых детерминированных языков.

Параграф 4 демонстрирует довольно интересный образчик взаимодействия теорий регулярных и бесконтекстных языков. В нем представлена теория Dғрафов, которые отвечают действующим в реальное время магазинным автоматам с одним поворотом. Эта теория без натяжки может быть названа алгебраической. В

ее основе лежит понятие модуля. В рассматриваемом случае привлечение алгебраических понятий и приемов оказалось возможным в той же мере, в какой это сделано в [Eilenberg 74] для изучения конечных автоматов. В частности, техника сравнения конечных автоматов, описанная в [Eilenberg 74], обобщается для проверки эквивалентности совершенных магазинных автоматов с одним поворотом, действующих в реальное время.

§1. Праволинейные магазинные автоматы и проблема регулярности

Вопрос регулярности является одним из наиболее важных в теории формальных языков и ее приложениях. В случае детерминированных языков он представляет особый интерес. Однако после работы [Stearns 67] проблема регулярности в этом случае практически не рассматривалась. Не появилось эффективных алгоритмов проверки регулярности даже для формализмов, описывающих подклассы детерминированных языков. Настоящая глава содержит достаточное условие регулярности бесконтекстного языка, проверяемое по (4,4)ядру магазинного автомата. В детерминированном случае для этой проверки можно взять несколько меньшее ядро. Здесь существенно используется понятие графа магазинного автомата.

Раздел 1.1 определяет понятие праволинейного магазинного автомата. Пометки парных циклов праволинейного магазинного автомата удовлетворяют ограничению, позволяющему только такое разрастание допускаемых цепочек, которое обеспечивается конечными автоматами или праволинейными грамматиками. Устанавливается, что праволинейный магазинный автомат допускает регулярное множество. Рассматривается признак праволинейности магазинного автомата, определяющий достаточное условие регулярности допускаемого языка. Формулируется алгоритм проверки праволинейности магазинного автомата, представляющий собой частичный алгоритм проверки регулярности бесконтекстного языка.

В разделе 1.2 приводится более "сильное"достаточное условие регулярности бесконтекстного языка, полученное в работе [Вылиток 98] путем обобщения результата, представленного в разделе 1.1. Точнее, в [Вылиток 98] расширяется понятие так называемого монома; это расширение позволяет выделить подкласс магазинных автоматов, который является аналогом определенного Н. Хомским класса грамматик без самовставления.

В разделе 1.3 указываются некоторые неясности работы [Stearns 67].

1.1. Мономы и праволинейность

Пусть маршрут T таков, что для некоторых T_1, T_2 тройка (T, T_1, T_2) является гнездом. Тогда назовем T *открывающим маршрутом*.

Заметим, что каждый непустой начальный участок открывающего маршрута является накапливающим.

Пусть T — читающий открывающий цикл. Пусть T_2 не является читающим стирающим циклом для любой четверки (T_0, T_1, T_2, T_3), такой, что $T_0TT_1T_2T_3$ есть маршрут, в котором T и T_2 парны. Тогда будем говорить, что цикл T есть *моном* (в маршруте $T_0TT_1T_2T_3$).

Пусть каждый читающий открывающий цикл магазинного автомата M является мономом. Тогда назовем автомат M *праволинейным*.

Заметим, что регулярность языка, допускаемого праволинейным автоматом, было бы нетрудно вывести из построения по магазинному автомату бесконтекстной грамматики, которое рассмотрено в [Вылиток -Станевичене -Чернцов 96] и [Вылиток 98] и явилось базой остальных приведенных в последней работе результатов. Упомянутое построение является обобщением алгоритма, который сформулирован в, например, [Ginsburg 66] и который строит по конечному автомату праволинейную грамматику. Оно позволяет легко установить синонимичность для целого ряда названий подклассов бесконтекстных языков, возникших независимо в теории автоматов и теории грамматик.

Но мы докажем регулярность языка, допускаемого праволинейным магазинным автоматом, опираясь на материал главы 1.

Теорема 1. Пусть M — праволинейный магазинный автомат. Тогда язык $L(M)$ регулярен.

Доказательство. Пусть, для определенности, входной алфавит автомата M есть Σ. Пусть $D = Graph(M)$.

Докажем, что КСвыражение $Expr(D)$, определенное в параграфе 3 главы 1, является псевдосогласующим. Так как псевдосогласующее КСвыражение задает регулярный язык, отсюда будет следовать регулярность языка $L(M) = L(D) = L(Expr(D))$.

Вспомним определенные в параграфе 3 главы 1 КСвыражение $\mathcal{E}(D)$, которое задает множество $Sentences(D)$, и морфизм φ, который переводит КСвыражение $\mathcal{E}(D)$ в $Expr(D)$. Заметим, что

$$\omega(\xi) \subseteq Sentences(D) \text{ для } \xi \in Clan(\mathcal{E}(D)),$$

а

$$\omega(T) \in L(D) \text{ для } T \in \omega(\xi).$$

Здесь толкование операции ω (пометка фракции или пометка маршрута) определяется видом ее операнда.

Индукцией по $k \geq 0$ докажем, что для нейтрального маршрута T длины $2k$ фракция

$$\xi_T = \varphi(schemes(T))$$

не является вставляющей.

При $k = 0$ маршрут T пуст, и фракция

$$\xi_T = ({}_{pair(T)}\epsilon)_{pair(T)}$$

не является вставляющей, поскольку даже циклов не содержит.

Пусть $k > 0$. Тогда для некоторых парных дуг π_1, π_2 и нейтральных маршрутов T_1, T_2 имеем (по определению схемы) равенства

$$T = \pi_1 T_1 \pi_2 T_2,$$

$$scheme(T) = ({}_{pair(T)}\pi_1 scheme(T_1)\pi_2 scheme(T_2))_{pair(T)}.$$

По предположению индукции фракции

$$\xi_{T_1} = \varphi(scheme(T_1))$$

и

$$\xi_{T_2} = \varphi(scheme(T_2))$$

не являются вставляющими. Допустим, что фракция

$$\xi_T = \varphi(scheme(T))$$

является вставляющей. Тогда в $scheme(T)$ есть парные $pair(T)$циклы, пометки которых (напомним, что это маршруты) обе несут цепочки из Σ^+. Следовательно, левый из этих циклов не есть моном в T, вопреки праволинейности автомата M. Сообразуясь с замечанием, сделанным выше, выводим из доказанного, что каждая фракция клана КСвыражения $Expr(D)$ не является всталяющей. Значит, $Expr(D)$ удовлетворяет определению псевдосогласующего КСвыражения $\square$

Лемма 1. Пусть M есть детерминированный магазинный автомат. Пусть дуги π_1, π_2 его маршрутов $T = T_1\pi_1T_2$, $T_1\pi_2$ различны. Пусть T есть накапливающий цикл. Тогда

$$\Lambda \neq \omega(\pi_1) \neq \omega(\pi_2) \neq \Lambda.$$

Доказательство. Из равенств

$$beg(\pi_1) = end(T_1) = beg(\pi_2)$$

и из детерминированности автомата следует, что либо π_1, π_2 различаются своими пометками, либо обе являются стирающими и различаются только своими конечными вершинами.

В первом случае из детерминированности следует, что $\omega(\pi_1)$ и $\omega(\pi_2)$ непусты, и лемма верна.

Во втором случае дуга π_1 имеет парную себе в $T_1\pi_1$, так как T есть накапливающий маршрут. Но тогда и дуга π_2 обязана иметь парную в $T_1\pi_2$ и, следовательно, $end(\pi_2) = end(\pi_1)$. Таким образом, $\pi_2 = \pi_1$, т.е. рассматриваемый случай не может возникнуть $\square$

Из леммы 1 следует, что открывающий цикл детерминированного магазинного автомата имеет непустую пометку. Действительно, если T — накапливающий цикл и маршрут TT' нейтрален, то T и некоторый начальный участок маршрута T' удовлетворяют условию леммы 1.

Следующая теорема означает, что для проверки детерминированного магазинного автомата M на праволинейность достаточно исследовать множество $Core(M, 3, 3)$.

Теорема 2. Если каждый читающий открывающий цикл каждого маршрута $T \in Core(M, 3, 3)$ есть моном, то детерминированный магазинный автомат M праволинеен.

Доказательство. Согласно определению праволинейного автомата достаточно доказать, что каждый читающий открывающий цикл в предложениях автомата M является мономом.

Пусть предложение T автомата M имеет вид

$$T = T_1\pi_1T_2T_3T_4\pi_2T_5\pi_3T_6,$$

где

$$(\pi_1T_2, T_3, T_4\pi_2T_5\pi_3)$$

есть гнездо, π_1T_2 и $T_7 = T_4\pi_2T_5\pi_3$ — циклы, π_1, π_2, π_3 — дуги и $\omega(\pi_2) \neq \Lambda$. Рассмотрим маршрут $T' = T'_1\pi_1T'_2T'_3T'_4\pi_2T'_5\pi_3T'_6 \in reduction(T_1,\ \pi_1,\ T_2'',\ beg(T_3),\ T_3'',\ end(T_3),\ T_4'',\ \pi_2,\ T_5'',\ \pi_3,\ T_6)$, где для $j \in \{2, 3, 4, 5\}$ маршрут T_j'' принадлежит

множеству $reduction(T_j)$. Из построения маршрута T' следует, что $(\pi_1 T'_2,\ T'_3,\ T'_7)$, где $T'_7 = T'_4\pi_2 T'_5\pi_3$, есть гнездо, $\pi_1 T'_2$ и T'_7 — циклы и пометка последнего цикла непуста. Убедимся, что T' является (3,3)каноном, т.е. что в T' не может быть больше трех последовательных нейтральных циклов и больше трех "концентрических"гнезд, образованных циклами и обладающих одними и теми же начальной и конечной вершинами.

Пусть участок T_0 маршрута T' является нейтральным циклом или гнездом, образованным парными циклами. Тогда, в силу построения маршрута T', T_0 содержит хотя бы один из участков, сохраняемых операцией $reduction$ при переработке T в T'. Если T_0 содержит дугу π_1 или π_3, то он содержит их обе, так как эти дуги парны. Если при этом T_0 является нейтральным циклом, то он не разбивается на два нейтральных цикла; иное противоречило бы построению T'_1 или T'_6. Такой нейтральный цикл допустим даже в (1,1)каноне. Если T_0 — гнездо, образованное парными циклами, то вследствие нейтральности T'_3 и парности дуг π_1, π_3 наиболее сложный из возможных случаев его устройства характеризуется равенством

$$T_0 = T_{11}T_{12}T_{13}T_{04}T_{23}T_{22}T_{21},$$

где для $1 \le j \le 3$ T_{1j} и T_{2j} — парные циклы, циклы T_{11} и T_{21} содержат соответственно парные друг другу дуги π_1 и π_3, T_{22} — дугу π_2, T_{13} или T_{23} имеет непустой общий участок с T'_3. Такое гнездо не противоречит определению (3,3)канона.

Пусть теперь T_0 не содержит дуг π_1 и π_3. Тогда этот маршрут не может иметь общих участков с T'_1 или T'_6 и расположен в

$$T'_2T'_3T'_4\pi_2T'_5.$$

Если T_0 является нейтральным циклом, то, по-прежнему, в худшем случае он разбивается на три нейтральных цикла, из которых один содержит пустой начальный участок маршрута T'_3, второй — пустой конечный участок этого маршрута, третий — дугу π_2. Если T_0 — гнездо, образованное парными циклами, то ни один из его циклов не содержится целиком в T'_3 (это противоречило бы нейтральности T'_3 или тому, что в T'_3 не может быть парных циклов). Следовательно, наиболее сложный случай устройства гнезда T_0 проще, чем рассмотренный выше, и характеризуется равенством

$$T_0 = T_{12}T_{13}T_{04}T_{23}T_{22},$$

где T_{1i} и T_{2i} — парные циклы для $i = 2, 3$; они характеризуются как в описанном выше случае.

Итак, T' является (3,3)каноном.

Из леммы 1 следует, что накапливающий цикл $\pi_1 T'_2$ гнезда $(\pi_1 T'_2,\ T'_3,\ T'_7)$ является читающим. Следовательно, так как T'_7 читающий, $\pi_1 T'_2$ не есть моном в каноне T', вопреки условию теоремы.

Вполне аналогично рассматривается случай, когда предложение T автомата M имеет вид

$$T = T_1\pi_1T_2T_3T_4\pi_2T_5,$$

где $(\pi_1 T_2,\ T_3,\ T_4\pi_2)$ есть гнездо, $\pi_1 T_2$ и $T_4\pi_2$ — циклы, π_1, π_2 — дуги и $\omega(\pi_2) \ne \Lambda$ □

Теорема 2 дает проверяемое по (3,3)ядру (см. алгоритм ниже) условие регулярности детерминированного языка. Это условие не является необходимым. В

самом деле, определенный в [Станевичене 95] однозначный детерминированный магазинный автомат допускает регулярное множество

$$\{u0v0 \mid u \in C^+, v \in \{a,b\}^*\},$$

где C есть алфавит n отличных от нуля символов для некоторого натурального n. Здесь мы приведем пример детерминированного магазинного автомата, который является самым, повидимому, лаконичным из неправолинейных однозначных совершенных детерминированных магазинных автоматов, допускающих регулярные множества. Автомат

$$(\{p_0,p_1,p_2,p_3,f\},\{a,b\},\{a,b,Z_0\},Z_0,\delta,p_0,\{f\}),$$

где множество δ состоит из команд

$(p_0,\Lambda,Z_0)\to(p_0,Z_0b)$,
$(p_0,a,b)\to(p_0,ba)$,
$(p_0,a,a)\to(p_0,aa)$,
$(p_0,b,a)\to(p_1,\Lambda)$,
$(p_1,a,a)\to(p_1,\Lambda)$,
$(p_1,b,b)\to(f,\Lambda)$,
$(p_1,b,a)\to(p_2,\Lambda)$,
$(p_2,\Lambda,a)\to(p_2,\Lambda)$,
$(p_2,\Lambda,b)\to(f,\Lambda)$,
$(p_1,a,b)\to(p_3,\Lambda)$,
$(p_3,a,Z_0)\to(p_3,Z_0a)$,
$(p_3,\Lambda,a)\to(p_3,\Lambda)$,
$(p_3,b,Z_0)\to(p_3,Z_0b)$,
$(p_3,\Lambda,b)\to(f,\Lambda)$,

допускает язык

$$\{a^mba^nb \mid \ m\geq 1, n\geq 0\}.$$

Каждое предложение этого автомата имеет одну из следующих форм:

(1) $m=1$:
  (a) $n>0$: $T_0\pi_1\pi_3C_1^{n-1}T_1$;
  (b) $n=0$: $T_0\pi_1\pi_2$;
(2) $m>1$:
  (a) $n\geq m$: $T_0C_2^{m-2}T_2C_3^{m-2}\pi_4\pi_3C_1^{n-m}T_1$;
  (b) $n=m-1$: $T_0C_2^{m-2}T_2C_3^{n-1}\pi_4\pi_2$;
  (c) $n=m-2$: $T_0C_2^{m-2}T_2C_3^{n}T_3$;
  (d) $0\leq n<m-2$: $T_0C_2^{m-2}T_2C_3^{n}\pi_7C_4^{m-n-3}T_4$,

где

$T_0=(p_0,Z_0)\frac{\Lambda}{+b}(p_0,b)\frac{a}{+a}(p_0,a)$, $\pi_1=(p_0,a)\frac{b}{-a}(p_1,b)$,
$\pi_2=(p_1,b)\frac{b}{-b}(f,Z_0)$, $\pi_3=(p_1,b)\frac{a}{-b}(p_3,Z_0)$,
$C_1=(p_3,Z_0)\frac{a}{+a}(p_3,a)\frac{\Lambda}{-a}(p_3,Z_0)$, $T_1=(p_3,Z_0)\frac{b}{+b}(p_3,b)\frac{\Lambda}{-b}(f,Z_0)$,
$C_2=(p_0,a)\frac{a}{+a}(p_0,a)$, $T_2=(p_0,a)\frac{a}{+a}(p_0,a)\frac{b}{-a}(p_1,a)$,
$C_3=(p_1,a)\frac{a}{-a}(p_1,a)$, $\pi_4=(p_1,a)\frac{a}{-a}(p_1,b)$,
$T_3=(p_1,a)\frac{b}{-a}(p_2,b)\pi_6$, $\pi_6=(p_2,b)\frac{\Lambda}{-b}(f,Z_0)$,
$T_4=(p_2,a)\frac{\Lambda}{-a}(p_2,b)\pi_6$, $\pi_7=(p_1,a)\frac{b}{-a}(p_2,a)$,
$C_4=(p_2,a)\frac{\Lambda}{-a}(p_2,a)$.

Здесь записи дуг используют в качестве разделителя вершин "дробь", в числителе которой пометка, а в знаменателе след дуги.

C_2 есть открывающий и читающий цикл, но не моном.

Алгоритм 1. Проверка праволинейности.

Вход. Детерминированный магазинный автомат M.

Выход. Если M праволинеен, то ответ "$L(M)$ регулярен", иначе "M имеет накапливающий цикл, который не есть моном".

Метод. Если каждый открывающий цикл любого маршрута из $Core(M,3,3)$ есть моном, то ответить "$L(M)$ регулярен", иначе ответить "M имеет накапливающий цикл, который не есть моном".

Из конечности множества $Core(M,3,3)$ и теоремы 2 следует, что алгоритм заканчивает работу и дает правильный ответ.

1.2. Согласующие и псевдосогласующие Dғрафы. Охарактеризуем еще одно средство задания регулярных языков — магазинные автоматы без самовставления. Они определены в [Вылиток 98] и включают праволинейные магазинные автоматы как собственный подкласс. Сформулируем нашу версию определения магазинного автомата без самовставления, проводя параллель с понятием псевдосогласующего КСвыражения.

Пусть какое-либо предложение Dғрафа D содержит гнездо, образованное парными циклами, каждый из которых имеет непустую пометку. Тогда назовем D *согласующим Dғрафом* . В противном случае будем называть Dғраф *псевдосогласующим.*

Пусть граф магазинного автомата M является псевдосогласующим Dғрафом. Тогда назовем M *магазинным автоматом без самовставления.*

Вопрос принадлежности магазинного автомата данному классу разрешается с помощью (4,4)ядра. Действительно, применяя прием, используемый в доказательстве теоремы 2, можно убедиться, что предложение

$$T_1\pi_1T_2\pi_2T_3T_4T_5\pi_3T_6\pi_4T_7,$$

в котором пометки дуг π_2 и π_3 непусты, а участки

$$\pi_1T_2\pi_2T_3 \text{ и } T_5\pi_3T_6\pi_4$$

являются парными циклами, редуцируется до (4,4)канона

$$T_1'\pi_1T_2'\pi_2T_3'T_4'T_5'\pi_3T_6'\pi_4T_7'$$

с парными циклами

$$\pi_1T_2'\pi_2T_3' \text{ и } T_5'\pi_3T_6'\pi_4.$$

Увеличение параметров ядра по сравнению с детерминированным случаем обусловлено тем, что для недетерминированных магазинных автоматов нет возможности опереться на непустоту пометки открывающего цикла (ср. использование леммы 1 в доказательстве теоремы 2).

1.3. Некоторые неясности одной работы о проверке регулярности детерминированного языка. Некоторые детали работы [Stearns 67] перекликаются

с особенностями нашего подхода. Однако попытка разобраться в ней наталкивается на некоторые трудности. Они начинаются уже с определений. Одним из важнейших в работе является понятие квазипустого слова. Его определение вряд ли подразумевает, что каждое ленточное слово, стираемое ε-движениями (этот и некоторые другие употребленные далее термины см. в [Stearns 67]), считается квазипустым. Действительно, рассмотрим пример магазинного автомата, определяемого дугами

$(p_0, Z_0)\frac{\Lambda}{+Z}(p_0, Z)$,
$(p_0, Z)\frac{\Lambda}{+Z}(p_1, Z)$,
$(p_1, Z)\frac{\Lambda}{-Z}(p_2, Z)$,
$(p_2, Z)\frac{\Lambda}{-Z}(f, Z_0)$.

Единственное предложение автомата имеет пустую пометку. ε-Движение

$$(p_1, Z) \uparrow (\Lambda)(p_2, \Lambda)$$

не приводит автомат в состояние, в котором он остался бы при стирании копии слова Z. Следовательно, слово Z не удовлетворяет определению квазипустого слова.

(Взяв нужное число состояний, пример можно изменить так, что длина успешного пути увеличится.)

Можно предположить, что автор хотел назвать квазипустыми слова, стираемые циклами с пустой пометкой (более того, такими циклами с пустой пометкой, которые имеют парный себе накапливающий цикл). Но тогда становится неясной теорема 3 работы [Stearns 67]. В самом деле, рассмотрим пример детерминированного магазинного автомата

$$M_n = (\{p_0, p_1, p_2\}, \{a, b\}, \{Z_0, \ldots, Z_n\}, Z_0, \delta, p_0, \{p_2\}),$$

допускающего (весьма нерационально) регулярный язык $\{a^m b \mid m \geq 1\}$:

1) $(p_0, \Lambda, Z_i) \to (p_0, Z_i Z_{i+1})$, $0 \leq i \leq n-1$,
2) $(p_0, a, Z_n) \to (p_0, Z_n Z_1)$,
3) $(p_0, b, Z_n) \to (p_1, \Lambda)$,
4) $(p_1, \Lambda, Z_i) \to (p_1, \Lambda)$, $1 \leq i \leq n$,
5) $(p_1, \Lambda, Z_0) \to (p_2, Z_0)$.

Команды 1, 2 соответственно каждому прочитанному a записывают в магазин цепочку

$$Z_2 \ldots Z_n Z_1.$$

Это квазипустое слово: начав стирать его в состоянии p_0, автомат переходит в состояние p_1, в котором и остается, пока не стерто все, кроме маркера дна.

При любом $n \geq 1$ автомат M_n имеет три состояния. Используемое в теореме 3 число $q! + 1$ есть $3! + 1 = 7$.

Ленточные слова автомата M_n имеют вид

$$Z_0 (Z_1 \ldots Z_n)^k \gamma,$$

где $k \geq 0$, а γ есть префикс цепочки $Z_1 \ldots Z_n$.

Пусть $n = 8$, $N = \{1, \ldots, 7\}$. Таким образом, множество N содержит $q! + 1$ различных целых чисел, меньших n. Рассмотрим $Z_1 \ldots Z_8$ — возможное из верхних в магазине слов. Пусть $e = 7$. Для любого числа $f \in N$, $f < e$, слово

$Z_{f+1} \dots Z_e$ не является квазипустым, так как $(Z_{f+1} \dots Z_e)^k$ для любого $k > 1$ не может входить в ленточное слово автомата и не может быть стерто автоматом.

Теорема 3 работы [Stearns 67] вызывает сомнение и без приведенного обсуждения: подозрительна возможность оценить сверху длину квазипустого слова, как бы оно ни определялось, только с помощью числа состояний, без учета числа магазинных символов. Повидимому, возможен какой-то аналог этой теоремы, использующий понятие вершины.

Заметим, что можно превратить автомат M_n в наполняющий магазин в реальное время заменой команд 1, 2 на команды

$$(p_0, a, Z_0) \to (p_0, ZZ_1 \dots Z_n), \ Z \in \{Z_0, Z_n\}.$$

При этом вид ленточных слов остается таким же. Можно строить и совершенные автоматы M'_n, наполняющие магазин в реальное время, перейдя от языка

$$\{a^m b \mid m \geq 1\}$$

к языкам

$$L_n = \{a^{mn} b \mid m \geq 1\}.$$

Здесь для каждого L_n число n определяет число магазинных символов автомата M'_n, отличных от маркера дна.

§2. Проблема построения графа сопряжений

Проблема эквивалентности детерминированных магазинных автоматов является ключевой в развитии некоторых направлений теории языков и автоматов. Естественна идея использовать имитацию одновременной работы двух детерминированных магазинных автоматов для проверки их эквивалентности. Еще в [Valiant 73] предложена техника имитации, применимая в некоторых подклассах детерминированных магазинных автоматов и использованная впоследствии в работах [Романовский 80, Романовский 86], предлагающих решения проблемы эквивалентности в классах детерминированных магазинных автоматов частного вида. (Впрочем, решение, представленное в [Романовский 86], ошибочно, что обсуждается в разделе 2 настоящего параграфа.)

Здесь рассматривается задача построения по двум эквивалентным детерминированным магазинным автоматам, заданным своими графами, некоторого нового конечного ориентированного графа, каждый путь которого является, по существу, сопоставлением вычислений автоматов над одной и той же цепочкой. Последний граф назван графом сопряжений (под сопряжением подразумевается одновременное прослеживание вычислений различных автоматов над одной и той же цепочкой). Граф сопряжений содержит всю информацию, характеризующую требования к запоминающей способности имитатора, и фактически является полноценным его представлением.

Для любой заданной пары детерминированных магазинных автоматов легко построить конечное множество, содержащее в себе все дуги соответствующего графа сопряжений. Однако, выделение подмножества дуг графа сопряжений оказывается алгоритмически неразрешимым даже в классе детерминированных магазинных автоматов, наполняющих магазин в реальное время и допускающих регулярные языки. Это доказывается с помощью детерминированных магазинных

автоматов специального вида, которые строятся по произвольному заданному случаю проблемы соответствия Поста. Оказывается, что существование алгоритма, который строит по двум таким детерминированным магазинным автоматам их граф сопряжений, означало бы также и существование алгоритма решения проблемы соответствия Поста, что противоречило бы известному факту об алгоритмической неразрешимости этой проблемы.

В разделе 1 определяется понятие графа сопряжений и доказывается теорема об алгоритмической неразрешимости проблемы построения графа сопряжений. Раздел 2 содержит модификацию рассмотренных в доказательстве этой теоремы специальных детерминированных магазинных автоматов, которая превращает их в детерминированные магазинные автоматы, действующие в реальное время (ДМАРВ) [Романовский 86]. С помощью новых специальных детерминированных магазинных автоматов доказывается, что из существования описанного в [Романовский 86] имитатора совместной работы двух ДМАРВ следует существование алгоритма решения проблемы соответствия Поста. Построения работы [Романовский 86], возможно, законны для некоторого подкласса ДМАРВ.

В последующих примерах магазинных автоматов удобно допускать в графе магазинного автомата дуги с пустым зарядом и с зарядом вида $+X_1 \ldots + X_m$, где $m > 1$, $X_i \in \Gamma$ для $i = 1, \ldots, m$. Дуга с пустым зарядом как бы совмещает в себе две последовательные дуги, из которых первая добавляет символ в магазин, а вторая стирает этот символ. Дугу с зарядом $+X_1 \ldots + X_m$, $m > 1$, можно считать сокращением последовательности дуг с зарядами $+X_1, \ldots, +X_m$.

2.1. Граф сопряжений. Представление дуги магазинного автомата парой ее вершин, пометкой и зарядом позволяет видеть нейтральные участки в записях маршрутов без обращения к Dмножеству данного автомата. В самом деле, с таким представлением дуг в маршруте дотошно протоколируется вид последовательных конфигураций автомата (ср. пример, рассмотренный в разделе 1.1). В данном разделе удобно воспользоваться именно таким представлением дуг.

Далее для $i = 1, 2$

$$M_i = (K_i, \Sigma, \Gamma_i, Z_{0i}, \delta_i, p_{0i}, F_i)$$

есть магазинные автоматы,

$\pi_i \in E(M_i)$,
$P_i \in V(M_i)$,
$\varepsilon \notin (\Sigma \cup \{\Lambda\}) \times (\{+, -\}\Gamma_i)$,
$\Psi_i = \{P\varepsilon P | P \in V(M_i)\}$,
$\psi_i \in \Psi_i$,
$E_i = \Psi_i \cup E(M_i)$,
$\xi_i \in E_i$,
$T_i = \pi_{i1} \ldots \pi_{im_i}$

есть успешный маршрут автомата M_i,

π_{ij} — дуга для $j = 1, \ldots, m_i$.

Элементы множества Ψ_i уподобим дугам (это "фиктивные"дуги: любой пустой участок некоторого маршрута, имеющий вершину P, может быть заменен последовательностью $(P\varepsilon P)^k$, $k > 0$, фиктивных дуг) и введем для них понятия

начальной и конечной вершин:

$$beg(P\varepsilon P) = end(P\varepsilon P) = P.$$

Фиктивные дуги послужат для "выравнивания"сопрягаемых маршрутов.

Пусть $\omega(\pi_1) = \omega(\pi_2)$. Тогда назовем упорядоченную пару (π_1, π_2) (M_1, M_2)-*биребром*.

Каждую из упорядоченных пар (π_1, ψ_2) и (ψ_1, π_2) назовем (M_1, M_2)-*моноребром*.

Введем общее для (M_1, M_2)биребер и (M_1, M_2)моноребер название (M_1, M_2)-*ребра*.

Граф, множество вершин которого есть $V(M_1) \times V(M_2)$, а множество дуг есть $\{((P_1, P_2), (Q_1, Q_2))$ с весом $(\xi_1, \xi_2) | (\xi_1, \xi_2)$ есть (M_1, M_2)ребро, для $i = 1, 2$ $beg(\xi_i) = P_i$ и $end(\xi_i) = Q_i\}$, обозначим через $Union(M_1, M_2)$.

Вершину графа $Union(M_1, M_2)$ назовем (M_1, M_2)-*узлом*.

Приставка (M_1, M_2) далее будет опускаться, если это не повлечет недоразумений.

Мы ввели особые термины для вершин и дуг графа

$$Union(M_1, M_2),$$

так как эта уловка, повидимому, облегчит обсуждение данного графа наряду с графами магазинных автоматов M_1, M_2.

Пусть T_1 и T_2 — предложения автоматов M_1 и M_2 соответственно. Пусть $\omega(T_1) = \omega(T_2)$. Тогда назовем (T_1, T_2)-*сопряжением* путь τ в графе $Union(M_1, M_2)$, получаемый по T_1 и T_2 следующими действиями:

$\tau :=$ [пустой путь с узлом $((p_{01}, Z_{01}), (p_{02}, Z_{02}))$];
$n_1 := 1$; $n_2 := 1$; (n_i — счетчик дуг маршрута T_i)
пока $n_1 \le m_1 \wedge n_2 \le m_2$, выполнять следующее:
  если $\omega(\pi_{1n_1}) = \omega(\pi_{2n_2})$,
  то $[\tau := \tau(\pi_{1n_1}, \pi_{2n_2})$; $n_1 := n_1 + 1$; $n_2 := n_2 + 1]$,
  иначе
    если $\omega(\pi_{1n_1}) = \Lambda$,
    то $[\tau := \tau(\pi_{1n_1}, beg(\pi_{2n_2}) \; \varepsilon \; beg(\pi_{2n_2}))$; $n_1 := n_1 + 1]$,
    иначе (в этом случае $\omega(\pi_{2n_2}) = \Lambda$)
      $[\tau := \tau(beg(\pi_{1n_1}) \; \varepsilon \; beg(\pi_{1n_1}), \pi_{2n_2})$; $n_2 := n_2 + 1]$;
пока $n_1 \le m_1$, выполнять присваивания
  $[\tau := \tau(\pi_{1n_1}, end(T_2) \; \varepsilon \; end(T_2))$; $n_1 := n_1 + 1]$;
пока $n_2 \le m_2$, выполнять присваивания
  $[\tau := \tau(end(T_1) \; \varepsilon \; end(T_1), \pi_{2n_2})$; $n_2 := n_2 + 1]$.

Назовем *графом сопряжений* и обозначим через

$$Team(M_1, M_2)$$

такой подграф графа $Union(M_1, M_2)$, что каждое его ребро является ребром некоторого сопряжения.

Сформулируем проблему построения графа сопряжений следующим образом:

построить алгоритм, который по произвольной паре (M_1, M_2) эквивалентных магазинных автоматов получает граф $Team(M_1, M_2)$.

Докажем, что данная проблема неразрешима даже в классе детерминированных магазинных автоматов, наполняющих магазин в реальное время и допускающих регулярные языки. Построим специальный класс магазинных автоматов, для которого из разрешимости проблемы построения графа сопряжений следует разрешимость проблемы соответствия Поста.

Пусть упорядоченные пары цепочек

$$(x_1, y_1), \ldots, (x_n, y_n),$$

где $n \geq 1$, определяют случай проблемы соответствия Поста в алфавите $\{a, b\}$. Пусть

$$\Sigma = \{a, b, 0\} \cup \{c_1, \ldots, c_n\}$$

и $|\Sigma| = n + 3$.

Рассмотрим язык

$$L = \{c_{i_p} \ldots c_{i_1} 0 z 0 \mid p \geq 1,\ 1 \leq i_j \leq n \text{ для } j = 1, \ldots, p,\ z \in \{a, b\}^*\}.$$

Язык L регулярен, так как он допускается конечным автоматом, определяемым следующими командами:

$(q_0, c) \to q_1,\ c \in \{c_1, \ldots, c_n\}$,
$(q_1, c) \to q_1,\ c \in \{c_1, \ldots, c_n\}$,
$(q_1, 0) \to q_2$,
$(q_2, s) \to q_2,\ s \in \{a, b\}$,
$(q_2, 0) \to q_3$.

Состояние q_3 является заключительным состоянием данного конечного автомата.

Легко видеть справедливость следующих равенств:

$$L = L_x \cup (L - L_x) = L_y(L - L_y),$$

где

$$L_x = \{c_{i_p} \ldots c_{i_1} 0 x_{i_1} \ldots x_{i_p} 0 \mid p \geq 1,\ 1 \leq i_j \leq n \text{ для } j = 1, \ldots, p\},$$
$$L - L_x = \{c_{i_p} \ldots c_{i_1} 0 z 0 \mid p \geq 1,\ 1 \leq i_j \leq n \text{ для } j = 1, \ldots, p,$$
$$z \in \{a, b\}^*,\ z \neq x_{i_1} \ldots x_{i_p}\},$$

а L_y определяется аналогично языку L_x с помощью вторых элементов пар, определяющих случай проблемы соответствия Поста.

Пусть u^R обозначает обращение цепочки u.

Определим детерминированный магазинный автомат

$$M_x = (K_x, \Sigma, \Gamma_x, Z_{0x}, \delta_x, p_{0x}, F_x)$$

следующим образом.

$$K_x = \{p_{0x}, p_{1x}, p_{2x}, p_{3x}, p_{4x}, f_x\},$$
$$\Gamma_x = \{Z_{0x}, a, b\},$$
$$F_x = \{f_x\},$$

множество δ_x образовано командами следующих четырнадцати серий, в которых $i = 1, \ldots, n$, $r, s \in \{a, b\}$:

(1) $(p_{0x}, c_i, Z_{0x}) \to (p_{0x}, Z_{0x} x_i^R)$,

(2) $(p_{0x}, c_i, s) \to (p_{0x}, sx_i^R)$,
(3) $(p_{0x}, 0, s) \to (p_{1x}, s)$,
(4) $(p_{1x}, s, s) \to (p_{1x}, \Lambda)$,
(5) $(p_{1x}, 0, Z_{0x}) \to (f_x, Z_{0x})$,
(6) $(p_{1x}, 0, s) \to (p_{4x}, s)$,
(7) $(p_{1x}, r, s) \to (p_{2x}, s)$, $r \neq s$,
(8) $(p_{2x}, r, s) \to (p_{2x}, s)$,
(9) $(p_{2x}, 0, s) \to (p_{4x}, s)$,
(10) $(p_{1x}, r, Z_{0x}) \to (p_{3x}, Z_{0x})$,
(11) $(p_{3x}, r, Z_{0x}) \to (p_{3x}, Z_{0x})$,
(12) $(p_{3x}, 0, Z_{0x}) \to (f_x, Z_{0x})$,
(13) $(p_{4x}, \Lambda, s) \to (p_{4x}, \Lambda)$,
(14) $(p_{4x}, \Lambda, Z_{0x}) \to (f_x, Z_{0x})$.

Автомат M_x допускает язык L, "выделяя"цепочки множества L_x среди остальных цепочек языка L. В самом деле, команды серий (1)–(5) допускают цепочки множества L_x: (1) и (2) накапливают в магазине цепочку $x_{i_p}^R \dots x_{i_1}^R$, (3) переводят автомат в состояние p_{1x} сопоставления входной цепочки с содержимым магазина, (4) прочитывают $x_{i_1} \dots x_{i_p}$, команда (5) переводит автомат в заключительное состояние, в котором данный автомат не может применить какую-либо команду. Команды (6) позволяют "не дочитать"последовательность $x_{i_1} \dots x_{i_p}$, следующую за префиксом $c_{i_p} \dots c_{i_1} 0$ входной цепочки. Команды серий (7), (8), (9) обеспечивают распознавание цепочек вида $c_{i_p} \dots c_{i_1} 0usv0$, где $s \in \{a, b\}$, u является, а us не является собственным префиксом цепочки $x_{i_1} \dots x_{i_p}$. Команды серий (10), (11), (12) обеспечивают распознавание цепочек вида $c_{i_p} \dots c_{i_1} 0 x_{i_1} \dots x_{i_p} v0$, где $v \in \{a, b\}^+$. Команды серии (13) вместе с командой (14) стирают в магазине не использованный (в случае применения команд (6) или (7)) суффикс цепочки $x_{i_1} \dots x_{i_p}$. Только последние стирающие команды не читают из входной цепочки, так что детерминированный магазинный автомат M_x удовлетворяет определению автомата, наполняющего магазин в реальное время.

Аналогично определим детерминированный магазинный автомат

$$M_y = (K_y, \Sigma, \Gamma_y, Z_{0y}, \delta_y, p_{0y}, F_y),$$

который допускает язык L, выделяя цепочки множества L_y среди остальных его цепочек. Таким образом, детерминированные магазинные автоматы M_x и M_y эквивалентны и допускают регулярный язык.

Лемма 2. Граф $Team(M_x, M_y)$ имеет ребро

$$(15)\ ((p_{1x}, Z_{0x})\frac{0}{\Lambda}(f_x, Z_{0x}),\ (p_{1y}, Z_{0y})\frac{0}{\Lambda}(f_y, Z_{0y}))$$

тогда и только тогда, когда рассматриваемый случай проблемы соответствия Поста имеет решение.

Доказательство. Пусть

$$\pi_w = (p_{1w}, Z_{0w})\frac{0}{\Lambda}(f_w, Z_{0w}) \text{ для } w \in \{x, y\}.$$

Тогда ребро $\tau = (\pi_x, \pi_y)$ и есть ребро (15). Пусть граф $Team(M_x, M_y)$ имеет ребро (15). Из построения автомата M_x следует, что дуга π_x является дугой успешного

маршрута автомата M_x тогда и только тогда, когда пометка этого маршрута имеет вид $c_{i_p} \dots c_{i_1} 0 x_{i_1} \dots x_{i_p} 0$ для некоторого $p \geq 1$. Аналогично, дуга π_y является дугой успешного маршрута автомата M_y тогда и только тогда, когда пометка этого маршрута имеет вид $c_{j_t} \dots c_{j_1} 0 y_{j_1} \dots y_{j_t} 0$ для некоторого $t \geq 1$.

Ребро τ, по определению графа сопряжений, входит в некоторый путь графа $Team(M_x, M_y)$, являющийся для некоторых успешных маршрутов T_x и T_y (T_x, T_y)-сопряжением.

Согласно определению сопряжения имеем равенство $\omega(T_x) = \omega(T_y)$. Это равенство означает, что верно равенство

$$c_{i_p} \dots c_{i_1} 0 x_{i_1} \dots x_{i_p} 0 = c_{j_t} \dots c_{j_1} 0 y_{j_1} \dots y_{j_t} 0.$$

Но последнее равенство возможно в том и только в том случае, если наборы $i_1, \dots, i_p$ и $j_1, \dots, j_t$ совпадают и $x_{i_1} \dots x_{i_p} = y_{i_1} \dots y_{i_p}$. Отсюда видим, что данный случай проблемы соответствия Поста имеет решение.

Пусть теперь данный случай проблемы соответствия Поста имеет решение, т.е. существуют $p \geq 1$ и набор $(i_1, \dots, i_p)$, такие, что $x_{i_1} \dots x_{i_p} = y_{i_1} \dots y_{i_p}$. Обозначим через z цепочку $c_{i_p} \dots c_{i_1} 0 x_{i_1} \dots x_{i_p} 0$. Тогда $z \in L_x \cap L_y$. Следовательно, для $w \in \{x, y\}$ существует успешный маршрут T_w автомата M_w, где $\omega(T_x) = \omega(T_y) = z$. Из построения автоматов M_x и M_y следует, что для $w \in \{x, y\}$ маршрут T_w заканчивается дугой π_w. Из определения сопряжения следует, что (T_x, T_y)єопряжение заканчивается ребром τ. Так как граф $Team(M_x, M_y)$ содержит все ребра сопряжений, он содержит и ребро τ $\square$

Отметим, что M_x и M_y относятся к магазинным автоматам с одним поворотом. Их записи в магазин предшествуют прочтению первого вхождения нуля во входную цепочку; затем возможны только стирания.

Теорема 3. Проблема построения графа сопряжений алгоритмически неразрешима в классе детерминированных магазинных автоматов с одним поворотом, наполняющих магазин в реальное время.

Доказательство. Достаточно убедиться в несуществовании алгоритма, который строит для произвольного случая проблемы соответствия Поста граф $Team(M_x, M_y)$. Предположим, что такой алгоритм существует. Тогда из леммы следует такой алгоритм решения проблемы соответствия: построить $Team(M_x, M_y)$; если $Team(M_x, M_y)$ имеет ребро (15), то решающая последовательность существует, иначе не существует. Однако, проблема соответстия Поста алгоритмически неразрешима. Следовательно, наше предположение неверно $\square$

2.2. Неясные места одной работы о проблеме эквивалентности ДМАРВ. Рассмотрим ДМАРВ

$$N_x = (K_x, \Sigma, \Gamma_x, Z_{0x}, \delta_x, p_{0x}, F_x),$$

где

$K_x = \{p_{0x}, p_{1x}, p_{2x}, f_x\}$,
$\Gamma_x = \{Z_{0x}, a, b\}$,
$F_x = \{f_x, g_x\}$,
δ_x образовано командами десяти серий, в которых $i = 1, \dots, n$, $r, s \in \{a, b\}$:
(16) $(p_{0x}, c_i, Z_{0x}) \to (p_{0x}, Z_{0x} x_i^R)$,

(17) $(p_{0x}, c_i, s) \to (p_{0x}, sx_i^R)$,
(18) $(p_x, 0, s) \to (p_{1x}, s)$,
(19) $(p_{1x}, s, s) \to (p_{1x}, \Lambda)$,
(20) $(p_{1x}, 0, Z_{0x}) \to (f_x, Z_{0x})$,
(21) $(p_{1x}, 0, s) \to (g_x, s)$,
(22) $(p_{1x}, r, s) \to (p_{2x}, sr)$, $r \neq s$,
(23) $(p_{2x}, r, s) \to (p_{2x}, sr)$,
(24) $(p_{2x}, 0, s) \to (g_x, s)$,
(25) $(p_{1x}, r, Z_{0x}) \to (p_{2x}, Z_{0x}r)$.

Автомат N_x эквивалентен автомату M_x предыдущего раздела и получен из M_x, в частности, отбрасыванием команд, стирающих в магазине при обработке входных цепочек из $L - L_x$.

Заметим, что вычисления автомата N_x над цепочками из L_x не используют команд вида (21)–(25). Цепочки из L_x и только они приводят автомат N_x в состояние f_x.

Аналогично определим ДМАРВ

$$N_y = (K_y, \Sigma, \Gamma_y, Z_{0y}, \delta_y, p_{0y}, F_y),$$

где множество δ_y разбивается на подмножества, аналогичные подмножествам (16)–(25) и далее упоминаемые под теми же номерами.

Если случай проблемы соответствия Поста, определяющий автоматы N_x и N_y, имеет решение $(i_1, \ldots, i_p)$, то, согласно теореме 2 работы [Романовский 86], существуют такие натуральное число t и функция замещения [Романовский 86] ρ_t, что гнездовой стековый автомат

$$A = A_{\rho_t}(N_x, N_y) = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F, \models, \vdash, |)$$

успешно имитирует вычисления автоматов N_x и N_y над цепочкой

(26) $c_{i_p} \ldots c_{i_1} 0 x_{i_1} \ldots x_{i_p} 0 \in L_x \cap L_y$.

Согласно замечанию выше эта имитация не может использовать переходов типов (см. стр. 21 в [Романовский 86]) (2)–(4) и (32), построенных с использованием наших команд типов (21)–(25). Заметим также, что A окончит данную имитацию в состоянии

$$(f_x, f_y, \gamma)$$

для некоторой $\gamma \in \Gamma^*$.

Рассмотрим гнездовой стековый автомат

$$B = (Q, \Sigma, \Gamma, \delta_B, q_0, Z_0, Q \cap \{(f_x, f_y, \gamma) | \gamma \in \Gamma^*\}, \models, \vdash, |),$$

который отличается от A множеством заключительных состояний и в котором δ_B по сравнению с δ не содержит множества переходов типа (номера типов см. в [Романовский 86]) (2)–(4), (32), построенных с использованием хотя бы одной команды типов (21)–(25).

Из построения гнездового стекового автомата B вытекает

Теорема 4. B допускает непустой язык тогда и только тогда, когда рассматриваемый случай проблемы соответствия Поста имеет решение □

Так как проблема пустоты алгоритмически разрешима в классе гнездовых стековых автоматов, теорема 4 свидетельствует об ошибочности представленного в работе [Романовский 86] алгоритма проверки эквивалентности ДМАРВ.

§3. Несколько примеров алгоритмических проблем

Сначала дадим примеры разрешимых алгоритмических проблем и укажем для них алгоритмы, использующие понятие ядра.

Теорема 5. Каждый из следующих вопросов разрешим.

(1) Пуст ли язык, допускаемый магазинным автоматом?
(2) Бесконечен ли такой язык?
(3) Является ли магазинный автомат праволинейным?

Доказательство. Без ограничения общности можно полагать, что магазинный автомат является совершенным. Покажем, что для ответа на перечисленные в теореме вопросы достаточно рассмотреть его $(w, d$ядро для $w, d \leq 4$.

1. $L(M) = \emptyset$ тогда и только тогда, когда $Core(M, 1, 1) = \emptyset$.

2. Теорема о развитии означает, что каждое предложение магазинного автомата M получается из элемента ядра $Core(M1, 1)$ вставлением циклов, имеющихся в элементах множества $Core(M, 2, 1)$. Таким образом, язык $L(M)$ бесконечен тогда и только тогда, когда среди предложений множества $Core(M, 2, 1)$ есть содержащее нейтральный читающий цикл или гнездо, образованное циклами, хотя бы один из которых является читающим.

3. Заметим, что, с одной стороны, по определениям каждый праволинейны магазинный автомат является автоматом без самовставления. С другой стороны, существуют магазинные автоматы без самовставления, не удовлетворяющие определению праволинейного магазинного автомата. Таков, например, автомат

$$(\{p_0, p_1, f\},\ \{a\},\ \{Z_0, a\},\ Z_0,\ \delta,\ p_0,\ \{f\}),$$

который допускает множество a^+; его множество δ состоит из команд
$(p_0, \Lambda, Z) \to (p_0, Za),\ Z \in \{Z_0, a\}$,
$(q, a, a) \to (p_1, \Lambda),\ q \in \{p_0, p_1\}$,
$p_1, \Lambda, Z_0) \to (f, Z_0)$.

Таким образом, праволинейные магазинные автоматы образуют собственный подкласс автоматов без самовставления. Однако ответ на вопрос о праволинейности некоторого автомата M не проще, чем ответ на вопрос о том, является ли M автоматом без самовставления, так как не удается упростить схему доказательства, представленную в разделе 1.2 выше. Пользуясь этой схемой, можно убедиться, что если недетерминированный магазинный автомат не является праволинейным, то обнаружение этого гарантируется (4,4)ядром автомата.

Если же магазинный автомат праволинеен, то в элементах его (4,4)ядра, как и во всех его предложениях, каждый открывающий читающий цикл является мономом.

Таким образом, магазинный автомат M праволинеен тогда и только тогда, когда в предложениях ядра $Core(M, 4, 4)$ каждый открывающий читающий цикл есть моном □

Следующее отношение эквивалентности подходит (ср. [Eilenberg 74]) для доказательства разрешимости проблемы эквивалентности конечных автоматов. Пусть $\mathcal{A} = (K, \Sigma, \delta, p_0, F)$ есть конечный автомат, $S_1, S_2 \subseteq K$. Множества состояний

S_1, S_2 назовем эквивалентными в том и только том случае, когда $L(S_1) = L(S_2)$, где

$$L(S) = \{x \in \Sigma^* \mid \exists p \in S \quad \exists f \in F \quad pxf \text{ есть путь на } \mathcal{A}\} \quad \forall S \subseteq K.$$

Понятие Dғрафа приводит к идее использования аналогичного отношения (определенного на множестве всех подмножеств вершин Dғрафа) для доказательства разрешимости проблемы эквивалентности магазинных автоматов некоторых подклассов. Ясно, что идея оправдана, когда магазинные автоматы допускают регулярные множества и, например, обладают каким-либо из следующих трех свойств.

(1) Автомат никогда не стирает в магазине.
(2) В любом гнезде хотя бы один из парных участков не является циклом.
(3) Автомат является праволинейным.

Рассмотрим проблему экивалентности множеств вершин в случае детерминированных магазинных автоматов.

Теорема 6. Проблема эквивалентности множеств вершин неразрешима в случае детерминированных магазинных автоматов, действующих в реальное время.

Доказательство. Построим специальный подкласс детерминированных магазинных автоматов, действующих в реальное время. Между автоматами подкласса и случаями проблемы соответствия Поста в некотором алфавите Σ будет иметь место взаимнооднозначное соответствие.

Автомат

$$M_{x_1,\ldots,x_n,y_1,\ldots,y_n} = (K, \Delta, \Gamma, Z_0, \delta, p_0, \{f\}),$$

где $x_1, \ldots, x_n, y_1, \ldots, y_n \in \Sigma^+$, $Z_0 \notin \Sigma$, $\Gamma = \Sigma \cup \{Z_0\}$, $\Delta = \Sigma \cup \{a, b, c, d_1, \ldots, d_n\}$ есть алфавит из $|\Sigma| + n + 3$ символов, определяется следующим образом.

Пусть $1 \le i \le n$, $Z \in \Gamma$, $g, h \in \Sigma$, w^R обозначает обращение цепочки w. Тогда множество команд δ задается следующими формулами:

1) $(p_0, a, Z_0) \to (p_1, Z_0)$,
2) $(p_0, b, Z_0) \to (p_2, Z_0)$,
3) $(p_1, d_i, Z) \to (p_1, Zx_i^R)$,
4) $(p_l, c, g) \to (q, g)$, $l = 1, 2$,
5) $(q, g, g) \to (q, \Lambda)$,
6) $(q, c, g) \to (f, g)$,
7) $(q, g, Z_0) \to (r, Z_0)$,
8) $(q, h, g) \to (r, g)$, $h \ne g$,
9) $(r, g, Z) \to (r, Z)$,
10) $(r, c, Z) \to (f, Z)$,
11) $(p_2, d_i, Z) \to (p_2, Zy_i^R)$,
12) $(p_0, c, Z_0) \to (p_3, Z_0)$,
13) $(p_3, d_i, Z_0) \to (p_4, Z_0)$,
14) $(p_4, d_i, Z_0) \to (p_4, Z_0)$,
15) $(p_4, c, Z_0) \to (r, Z_0)$.

Данный автомат допускает язык $aL_1 \cup bL_2 \cup cL$, где

$$L = \{d_1, \ldots, d_n\}^+ c\Sigma^* c,$$

$$L_1 = L \setminus \{d_{i_k} \ldots d_{i_1} c x_{i_1} \ldots x_{i_k} c \mid \ k > 0, \ 1 \le i_j \le n \text{ для } 1 \le j \le k\},$$

$$L_2 = L \setminus \{d_{i_k} \dots d_{i_1} c y_{i_1} \dots y_{i_k} c \mid \ k > 0,\ 1 \le i_j \le n \text{ для } 1 \le j \le k\}.$$

Действительно, если в начальном состоянии прочитан символ c (см. команду 12), то далее применяются команды 13–15 и 9–10, ничего не запоминающие в магазине, т.е. определяющие, по существу, детерминированный конечный автомат. Они обеспечивают переход из состояния p_3 в заключительное состояние f под действием цепочки из L и только такой цепочки. Команды 1 и 2 "запускают подавтоматы", определяемые соответственно подмножествами команд $\{3, \dots, 10\}$ и $\{4, \dots, 10, 11\}$. Команды 3–10 допускают язык L_1 таким же способом, каким автомат M_x параграфа 2 допускает язык $L \setminus L_x$. Аналогично, однотипно построенные команды 11 и 4–10 допускают язык L_2.

Легко видеть, что $L = L_1 \cup L_2$ тогда и только тогда, когда

$$x_{i_1} \dots x_{i_k} \neq y_{i_1} \dots y_{i_k} \quad \forall (k > 0, 1 \le i_1, \dots, i_k \le n).$$

Следовательно, множества $\{(p_1, Z_0), (p_2, Z_0)\}$ и $\{(p_3, Z_0)\}$ эквивалентны тогда и только тогда, когда отвечающий автомату случай проблемы соответствия Поста не имеет решений $\square$

Идею, использованную в параграфе 2 для построения специальных детерминированных магазинных автоматов M_x и M_y, можно приспособить для доказательства неразрешимости вопроса, регулярно ли объединение двух непустых детерминированных языков. Точнее, мы докажем

Утверждение 1. Не существует алгоритма, который по любым заданным конечному автомату $\mathcal{A}$ и допускающим непустые языки детерминированным магазинным автоматам M_1 и M_2 определяет, верно ли равенство

$$L(M_1) \cup L(M_2) = L(\mathcal{A}).$$

Доказательство. Рассмотрим множество пар Dғрафов (D_x, D_y), биективно отображающееся на множество всех случаев проблемы соответствия Поста в алфавите $\{a, b\}$.

Пусть Σ, L, L_x и L_y определены, как в параграфе 2. Dғраф D_x определим как граф детерминированного магазинного автомата M'_x, получаемого из M_x добавлением еще одного состояния g_x, которое не является заключительным, и заменой команды (5) на

$$(p_{1x}, 0, Z_{0x}) \to (g_x, Z_{0x}).$$

Автомат M'_x допускает язык $L \setminus L_x$.

Аналогично определим автомат M'_y, допускающий язык $L \setminus L_y$.

Для каждой пары магазинных автоматов M'_x и M'_y рассмотрим один и тот же детерминированный конечный автомат $\mathcal{A}$, допускающий язык L; он определен в параграфе 2.

Ясно, что объединение $L(D_x) \cup L(D_y)$ совпадает с L тогда и только тогда, когда рассматриваемый случай проблемы соответствия Поста не имеет решений. Следовательно, не существует алгоритма, который для произвольного случая проблемы соответствия Поста определяет, справедливо ли равенство

$$L(D_x) \cup L(D_y) = L.$$

Отсюда выводим истинность утверждения $\square$

Видоизменение проблемы может оказаться плодотворным при поиске ее решений, которые могли бы конкурировать с известными по легкости применения. В связи с этим используем понятие Dғрафа для получения формулировок проблем эквивалентности и регулярности в терминах МПғпреобразований, определенных на детерминированных языках реального времени, являющихся пересечениями Dязыков с локальными множествами. Предварительно докажем две леммы.

Лемма 3. Язык успешных маршрутов Dғрафа является детерминированным языком реального времени.

Доказательство. Рассмотрим произвольный Dғраф

$$D = (\Sigma,\ V,\ \mathcal{P},\ \lambda,\ P_0,\ F).$$

Модифицируем алгоритм 2 главы 1 так, чтобы его выходом был магазинный автомат, допускающий не $L(D)$, а $Sentences(D)$. Для этого достаточно использовать саму дугу там, где алгоритм использовал ее пометку.

Обозначим автомат над алфавитом $E(D)$, производимый данным модифицированным алгоритмом, через $Super(D)$.

Из конструкции автомата $Super(D)$ следует, что он является детерминированным и что каждая его команда читает символ входного алфавита □

Заметим, что равносильными лемме 3 являются утверждения "язык успешных маршрутов магазинного автомата является детерминированным языком реального времени "и "пересечение подмножества Dязыка с локальным множеством является детерминированным языком реального времени".

Лемма 4. Для любого бесконтекстного языка L существует преобразователь с магазинной памятью S, согласованный с некоторым детерминированным магазинным автоматом M, действующим в реальное время, и такой, что $S(L(M)) = L$.

Доказательство. По доказанному в главе 1 бесконтекстный язык L определяется некоторым Dғрафом D. Из доказательства леммы 3 следует, что язык $Sentences(D) = L(Super(D))$ допускается некоторым детерминированным магазинным автоматом M, действующим в реальное время. Рассмотрим согласованный с автоматом M преобразователь $Meta_D$ с магазинной памятью, который соответственно входному символу π (дуге исходного Dғрафа) пишет на выходной ленте символ $\omega(\pi)$. Тогда

$$Meta_D(L(M)) = L(D) = L$$

□

Отметим теперь, что Dғрафы D_1 и D_2 эквивалентны тогда и только тогда, когда

$$Meta_{D_1}(L(Super(D_1))) = Meta_{D_2}(L(Super(D_2))),$$

и что язык, определяемый Dғрафом D, регулярен тогда и только тогда, когда регулярен язык

$$Meta_D(L(Super(D))).$$

Таким образом, эффективные решения частных случаев проблем эквивалентности и регулярности могут быть найдены и на пути исследования преобразователей определенного здесь вида.

§4. О магазинных автоматах с одним поворотом, действующих в реальное время

В данной главе развита теория Dғрафов, которые отвечают действующим в реальное время магазинным автоматам с одним поворотом. Пусть

$$D = (\Sigma,\ V,\ \mathcal{P},\ \lambda,\ P_0,\ F)$$

есть граф данного класса. Пусть

$$Turn = \{P \in V \mid \exists(\pi_1 \in Left(\mathcal{P}), \pi_2 \in Right(\mathcal{P}))$$

$$end(pi_1) = beg(pi_2) = P\}.$$

Менее формально, в $Turn$ входит общая вершина двух дуг, образующих нейтральный маршрут.

Графу D можно сопоставить два конечных автомата:

$$Push(D) = (\{beg(\pi), end(\pi) \mid \pi \in Left(\mathcal{P})\},\ \Sigma,\ Left(\mathcal{P}),\ p_0,\ Turn)$$

и

$$Pop(D) = (\{beg(\pi), end(\pi) \mid \pi \in Right(\mathcal{P})\},\ \Sigma,\ Right(\mathcal{P}),\ F,\ Turn).$$

Язык $L(Push(D))$ включает в себя множество цепочек, прочитываемых соответствующим графу магазинным автоматом до “поворота", т. е. перехода от накопления в магазине к стиранию. $L(Pop(D))$ включает множество обращений цепочек, прочитываемых после поворота. Механизм зарядов маршрутов позволяет выбрать из $L(Push(D))$ и $L(Pop(D))$ пары (x, y) цепочек, таких, что сцепления xy^R образуют язык $L(D)$. (Здесь y^R обозначает обращение цепочки y.)

Обозначим через $\mathcal{M}$ класс всех совершенных магазинных автоматов, действующих в реальное время (СМАРВ), а через $\mathcal{M}_1$ класс всех СМАРВ с одним поворотом. Описанное наблюдение приводит к мысли применить для исследования автоматов из $\mathcal{M}_1$ методы и приемы, разработанные для конечных автоматов. В [Eilenberg 74] особенно тщательно и полно изложены касающиеся конечных автоматов результаты, полученные на алгебраической основе. Множество Q состояний (или вершин) детерминированного конечного автомата, допускающего некоторое подмножество множества Σ^*, названо в [Eilenberg 74] правым Σ-модулем. На множестве Q задаются “действия"частичной функцией $\Theta : Q \times \Sigma^* \to Q$. Эта функция получается чисто алгебраическим расширением частичной функции $Q \times \Sigma \to Q$, которая определяет дуги рассматриваемого конечного автомата. В терминах действий изящно доказывается, например, разрешимость проблемы эквивалентности конечных автоматов.

Здесь мы обобщаем указанную технику в целях исследования автоматов из $\mathcal{M}_1$. Если по [Eilenberg 74] действие определяет некоторый путь в конечном автомате, то в нашем случае действие определяет два парных маршрута T_1 и T_2, переводя пару вершин

$$(beg(T_1),\ end(T_2))$$

в пару

$$(end(T_1),\ beg(T_2)).$$

В терминах обобщенных действий характеризуется язык, допускаемый автоматом из $\mathcal{M}_1$, и рассматриваются алгоритмы, доказывающие разрешимость проблемы эквивалентности в классе $\mathcal{M}_1$. Представляется правдоподобным, что некоторая комбинация "действий", законных для конечных автоматов, и "действий"в нашей трактовке позволит удачно обрабатывать более широкие классы магазинных автоматов.

Пусть

$$M = (K, \Sigma, \Gamma, Z_0, \delta, p_0, F)$$

есть автомат из $\mathcal{M}_1$. Пусть $V(M)$ — множество вершин автомата M, $\mathcal{R}(M) = V(M) \times V(M)$; $\Sigma_(= Left(\mathcal{P}(M))$, $\Sigma_) = Right(\mathcal{P}(M))$.

Так как автомат M является совершенным, любое его предложение является цепочкой Dязыка над Dмножеством $\mathcal{P}(M)$.

Пусть

$$\Psi = \{(s,t) \in \Sigma^* \times \Sigma^* \mid |s| = |t|\}.$$

Обозначим через ε элемент $(\Lambda, \Lambda) \in \Psi$. Определим на Ψ умножение следующим образом. Пусть $\psi_i = (s_i, t_i) \in \Psi$ для $i = 1, 2$. Тогда

$$\psi_1\psi_2 = (s_1 s_2, t_2 t_1).$$

(Заметим, что Ψ является моноидом с единицей ε.)

В следующем определении X, X_1, $X_2 \subseteq \mathcal{R}(M)$; a_1, $a_2 \in \Sigma$; P_1, P_2, Q_1, $Q_2 \in V(M)$; ψ, ψ_1, $\psi_2 \in \Psi$.

Определим функцию

$$\Delta_M : 2^{\mathcal{R}(M)} \times \Psi \to 2^{\mathcal{R}(M)},$$

которая задает *действия* на множестве всех подмножеств пар вершин автомата $M \in \mathcal{M}_1$, таким образом:

$\Delta_M(\Delta_M(X, \psi_1), \psi_2) = \Delta_M(X, \psi_1\psi_2)$,

$\Delta_M(X, \varepsilon) = X$,

$\Delta_M(X_1 \cup X_2, \psi) = \Delta_M(X_1, \psi) \cup \Delta_M(X_2, \psi)$,

$\Delta_M(\emptyset, \psi) = \emptyset$,

$\Delta_M(\{(P_1, Q_2)\}, (a_1, a_2)) = \{(Q_1, P_2) \mid \exists(\pi_1, \pi_2) \in \mathcal{P}(M)$: для $i = 1, 2$ $\omega(\pi_i) = a_i, beg(\pi_i) = P_i, end(\pi_i) = Q_i\}$.

Для сокращения записи применим следующие обозначения. Пусть X, $Y \subseteq \mathcal{R}(M)$, $\psi = (s_1, s_2) \in \Psi$, $Y = \Delta_M(X, \psi)$. Тогда запишем $Y = X\psi$ и $Y = X\langle s_1, s_2\rangle$. Если $X = \{(P_1, P_2)\}$, то будем применять также запись $Y = \langle P_1 s_1, s_2 P_2\rangle$.

Из данного определения видно, что функция Δ_M определяется множеством $\mathcal{P}(M)$. Согласно следующей лемме, это множество совпадает со множеством

$$P_{Core}(M) = \{(\pi_1, \pi_2) \in \Sigma_(\times \Sigma_) \mid$$

дуги π_1 и π_2 парны в некотором предложении из $Core(M, 1, 1)\}$.

Таким образом, определение функции Δ_M является конструктивным.

Лемма 5. Пусть $T = T_0\pi_1 T_1 \pi_2 T_2$ — предложение совершенного магазинного автомата M. Пусть дуги π_1 и π_2 парны в T. Тогда найдется предложение $T' = T'_0\pi_1 T'_1 \pi_2 T'_2 \in Core(M)$, в котором дуги π_1, π_2 парны.

Доказательство. Заметим, что маршрут T_1 нейтрален, так как дуги π_1 и π_2 парны.

Рассмотрим маршрут

$$T' = T'_0\pi_1 T'_1\pi_2 T'_2 \in reduction(T_0, \pi_1, T_1, \pi_2, T_2).$$

Маршрут T'_1, полученный редуцированием нейтрального маршрута T_1, является нейтральным. Таким образом, π_1 и π_2 парны в T'.

Убедимся, что T' является каноном. Заметим, что T'_0, T'_1 и T'_2 не содержат нейтральных и парных циклов. Следовательно, в T' каждый участок T_3, который является нейтральным циклом или гнездом, образованным парными циклами, содержит какую-либо из дуг π_1, π_2. Но тогда T_3 содержит обе эти дуги, так как они парны, а T_3 нейтрален. Отсюда выводим, что если T_3 — нейтральный цикл, то он не разбивается на два (и тем более на три) нейтральных цикла, а если T_3 — гнездо, то это простое гнездо. Таким образом, T' удовлетворяет определению (1,1)канона $\square$

Пусть далее везде, где это не оговаривается специально, под автоматом M понимается автомат $(K, \Sigma, \Gamma, Z_0, \delta, p_0, F)$ из $\mathcal{M}_1$.

Назовем пару $(P, P) \in \mathcal{R}(M)$ *терминальной парой вершин.*

Назовем *языком пары вершин* $(P, Q) \in \mathcal{R}(M)$ следующее множество цепочек: $L(P, Q) = \{s_1 s_2 \mid$ множество $\langle Ps_1, s_2 Q\rangle$ содержит хотя бы одну терминальную пару вершин$\}$.

Из этих определений следует, что для любого автомата M

$$L(M) = \bigcup_{f \in F} L((p_0, Z_0),\ (f, Z_0)).$$

Для любого элемента $\psi = (s_1, s_2) \in \Psi$ множество

$$\bigcup_{f \in F} \langle (p_0, Z_0)s_1, s_2(f, Z_0)\rangle$$

может быть вычислено за конечное число шагов. Это значит, что для автоматов из $\mathcal{M}_1$ можно решать проблему принадлежности с помощью следующего алгоритма.

Алгоритм 2. Проверка допустимости цепочки.

Вход. Автомат M, цепочка s.

Выход. Если $s \in L(M)$, то “да", иначе “нет".

Метод. Если длина цепочки s нечетна, то “нет";
если $s = \Lambda$, то если $p_0 \in F$, то “да", иначе “нет";
построить множество

$$X = \bigcup_{f \in F} \langle (p_0, Z_0)s_1, s_2(f, Z_0)\rangle,$$

где $s_1 s_2 = s$, $|s_1| = |s_2|$; если X содержит хотя бы одну терминальную пару вершин, то “да", иначе “нет".

Определим некоторые понятия, двойственные к уже введенным. Пусть X, X_1, $X_2 \subseteq \mathcal{R}(M)$; a_1, $a_2 \in \Sigma$; P_1, P_2, Q_1, $Q_2 \in V(M)$; ψ, ψ_1, $\psi_2 \in \Psi$.

Определим функцию $\Delta_M^{-1} : 2^{\mathcal{R}(M)} \times \Psi \to 2^{\mathcal{R}(M)}$ следующим образом:
$\Delta_M^{-1}(\Delta_M^{-1}(X, \psi_1), \psi_2) = \Delta_M^{-1}(X, \psi_1\psi_2)$,
$\Delta_M^{-1}(X, \varepsilon) = X$,

$\Delta_M^{-1}(X_1 \cup X_2, \psi) = \Delta_M^{-1}(X_1, \psi) \cup \Delta_M^{-1}(X_2, \psi)$,
$\Delta_M^{-1}(\emptyset, \psi) = \emptyset$,
$\Delta_M^{-1}(\{(P_1, Q_1)\}, (a_1, a_2)) = \{(P_2, Q_2) \mid (P_1, Q_1) \in \langle P_2 a_1, a_2 Q_2 \rangle\}$.

Введем сокращения, аналогичные принятым для Δ_M. Пусть X, $Y \subseteq \mathcal{R}(M)$, $\psi = (s_1, s_2) \in \Psi$, $Y = \Delta_M^{-1}(X, \psi)$. Тогда будем писать $Y = X^{-1}\psi$ и $Y = X^{-1}\langle s_1, s_2 \rangle$. Если $X = \{(P_1, P_2)\}$, то также возможна запись $Y = \langle P_1 s_1, s_2 P_2 \rangle^{-1}$.

Из определений Δ_M и Δ_M^{-1} вытекает справедливость следующей леммы.

Лемма 6. Пусть X, $Y \subseteq \mathcal{R}(M)$; X, $Y \neq \emptyset$; $\psi \in \Psi$. Тогда, если $Y \subseteq X\psi$, то $Y^{-1}\psi \cap X \neq \emptyset$, и если $X \subseteq Y^{-1}\psi$, то $X\psi \cap Y \neq \emptyset$ $\square$

Назовем *обратным языком пары вершин* $(P, Q) \in \mathcal{R}(M)$ следующее множество:

$$L^{-1}(P, Q) = \{s_1 s_2 \mid \exists f \in F : ((p_0, Z_0), (f, Z_0)) \in \langle P s_1, s_2 Q \rangle^{-1}\}.$$

Утверждение 2. $L(M) = \bigcup\limits_{(P,P) \in \mathcal{R}(M)} L^{-1}(P, P)$.

Доказательство. Сначала докажем соотношение

$$L(M) \subseteq \bigcup_{(P,P) \in \mathcal{R}(M)} L^{-1}(P, P).$$

По определению функции Δ_M для любой цепочки $s \in L(M)$, $s = s_1 s_2$, $|s_1| = |s_2|$, найдутся такая терминальная пара вершин $(P, P) \in \mathcal{R}(M)$ и такое заключительное состояние $f \in F$, что $(P, P) \in \langle (p_0, Z_0) s_1, s_2 (f, Z_0) \rangle$. По лемме 6 это значит, что

$$((p_0, Z_0), (f, Z_0)) \in \langle P s_1, s_2 P \rangle^{-1}.$$

Следовательно,

$$s_1 s_2 \in L^{-1}(P, P).$$

Таким образом,

$$\forall s \in L(M)\ \exists (P, P) \in \mathcal{R}(M) : s \in L^{-1}(P, P).$$

А это означает, что

$$L(M) \subseteq \bigcup_{(P,P) \in \mathcal{R}(M)} L^{-1}(P, P).$$

Теперь докажем, что

$$\bigcup_{(P,P) \in \mathcal{R}(M)} L^{-1}(P, P) \subseteq L(M).$$

Пусть

$$(P, P) \in \mathcal{R}(M),$$

$$s \in L^{-1}(P, P),\ s = s_1 s_2,\ |s_1| = |s_2|.$$

По определению функции Δ_M^{-1} найдется такое состояние $f \in F$, что

$$((p_0, Z_0), (f, Z_0)) \in \langle P s_1, s_2 P \rangle^{-1}.$$

Следовательно, в силу леммы 6

$$(P, P) \in \langle (p_0, Z_0) s_1, s_2 (f, Z_0) \rangle.$$

А это значит, что

$$s_1 s_2 \in L(M),$$

т. е.

$$\forall (P,P) \in \mathcal{R}(M)\ \forall s \in L^{-1}(P,P)\ s \in L(M).$$

Следовательно,

$$\bigcup_{(P,P)\in\mathcal{R}(M)} L^{-1}(P,P) \subseteq L(M)$$

и, в силу доказанного ранее,

$$L(M) = \bigcup_{(P,P)\in\mathcal{R}(M)} L^{-1}(P,P)$$

□

Пусть $X \in 2^{\mathcal{R}(M)}$. Назовем *языком множества пар вершин* объединение

$$L(X) = \bigcup_{(P,Q)\in X} L(P,Q).$$

Пусть

$$L(X)_k = \{s_1 s_2 \in L(X) \mid |s_1| = |s_2| \le k\}.$$

Назовем $L(X)_k$ *k-подмножеством* языка $L(X)$.

Пусть $L(X_1)_k = L(X_2)_k$, где $X_1,\ X_2 \in 2^{\mathcal{R}(M)}$, $k \ge 0$. Тогда будем называть множества X_1 и X_2 *k-неразличимыми* и писать

$$X_1 \equiv_k X_2.$$

Если X_1 и X_2 kнеразличимы для любого $k \ge 0$, то будем говорить, что они *неразличимы*, и писать

$$X_1 \equiv X_2.$$

Отношения $\equiv_k$ и $\equiv$ назовем отношениями *k-неразличимости* и *неразличимости* соответственно.

Лемма 7. Пусть $X \in 2^{\mathcal{R}(M)}$. Справедливо следующее соотношение:

$$L(X)_{k+1} = L(X)_k \cup \Big(\bigcup_{\substack{a,b\in\Sigma \\ L(X\langle a,b\rangle)_k \neq \emptyset}} aL(X\langle a,b\rangle)_k b \Big).$$

Доказательство. По определению языка множества пар вершин язык $L(X)_{k+1}$ состоит из всех цепочек множества $L(X)$, длина которых не превосходит $2(k+1)$. Цепочки множества $L(M)$, длина которых не превосходит $2k$, по определению образуют множество $L(X)_k$. Убедимся, что

$$\bigcup_{\substack{a,b\in\Sigma \\ L(X\langle a,b\rangle)_k \neq \emptyset}} aL(X\langle a,b\rangle)_k b$$

содержит все цепочки из $L(X)$, имеющие длину $2(k+1)$. Действительно:

$$\bigcup_{\substack{a,b\in\Sigma \\ L(X\langle a,b\rangle)_k \neq \emptyset}} aL(X\langle a,b\rangle)_k b = \bigcup_{\substack{a,b\in\Sigma \\ (P,Q)\in X \\ L(\langle Pa,bQ\rangle)_k \neq \emptyset}} aL(\langle Pa,bQ\rangle)_k b =$$

$$= \bigcup_{\substack{a,b\in\Sigma \\ (P,Q)\in X \\ L(\langle Pa,bQ\rangle)_k\neq\emptyset}} L(P,Q)_{k+1} =$$

$$= \{s_1s_2 \in \bigcup_{(P,Q)\in X} L(P,Q) \;\big|\; |s_1| = |s_2| \le k+1\} =$$

$$= \{s_1s_2 \in L(X) \;\big|\; |s_1| = |s_2| \le k+1\}$$

□

Лемма 8. Пусть $k \ge 1$; X_1, $X_2 \in 2^{\mathcal{R}(M)}$. Тогда

$$X_1 \equiv_{k+1} X_2 \Leftrightarrow X_1 \equiv_k X_2 \;\; \forall a,b \in \Sigma \; L(X_1\langle a,b\rangle)_k = L(X_2\langle a,b\rangle)_k.$$

Доказательство. Пусть $X_1 \equiv_{k+1} X_2$. По определению отношения k-неразличимости $L(X_1)_{k+1} = L(X_2)_{k+1}$. Следовательно, $L(X_1)_k = L(X_2)_k$. А это значит, что $X_1 \equiv_k X_2$. Докажем, что

$$\forall a,b \in \Sigma \; L(X_1\langle a,b\rangle)_k = L(X_2\langle a,b\rangle)_k.$$

Допустим противное. Тогда

$$\exists a,b \in \Sigma : L_1 = L(X_1\langle a,b\rangle)_k \ne L(X_2\langle a,b\rangle)_k = L_2.$$

Это значит, что найдется цепочка, принадлежащая одному языку и не принадлежащая другому. Пусть, например, существует такая цепочка $s \in \Sigma^*$, что $s \in L_1$ и $s \notin L_2$. Так как $s \in L_1 = L(X_1\langle a,b\rangle)_k$, то $asb \in L(X_1)_{k+1}$. Но по условию $X_1 \equiv_{k+1} X_2$, следовательно, $L(X_1)_{k+1} = L(X_2)_{k+1}$. Отсюда получаем, что $asb \in L(X_2)_{k+1}$, а это значит, что $s \in L(X_2\langle a,b\rangle)_k = L_2$. Но s была выбрана как цепочка, не принадлежащая L_2. Полученное противоречие доказывает, что

$$X_1 \equiv_{k+1} X_2 \Rightarrow X_1 \equiv_k X_2 \;\; \forall a,b \in \Sigma \; L(X_1\langle a,b\rangle)_k = L(X_2\langle a,b\rangle)_k.$$

Пусть теперь $X_1 \equiv_k X_2$ и $L(X_1\langle a,b\rangle)_k = L(X_2\langle a,b\rangle)_k$ для любых a, b из алфавита Σ. По лемме 7

$$L(X_1)_{k+1} = L(X_1)_k \cup (\bigcup_{\substack{a,b\in\Sigma \\ L(X_1\langle a,b\rangle)_k\neq\emptyset}} aL(X_1\langle a,b\rangle)_k b).$$

По условию $L(X_1\langle a,b\rangle)_k = L(X_2\langle a,b\rangle)_k$ для любых a, b из Σ и $X_1 \equiv_k X_2$. Последнее означает, что $L(X_1)_k = L(X_2)_k$. Следовательно,

$$L(X_1)_{k+1} = L(X_2)_k \cup (\bigcup_{\substack{a,b\in\Sigma \\ L(X_2\langle a,b\rangle)_k\neq\emptyset}} aL(X_2\langle a,b\rangle)_k b).$$

Пользуясь леммой 7, получаем, что $L(X_1)_{k+1} = L(X_2)_{k+1}$, т. е. $X_1 \equiv_{k+1} X_2$. Таким образом,

$$X_1 \equiv_k X_2 \;\; \forall a,b \in \Sigma \qquad L(X_1\langle a,b\rangle)_k = L(X_2\langle a,b\rangle)_k \Rightarrow X_1 \equiv_{k+1} X_2$$

□

Лемма 9. Пусть $k \ge 0$. Тогда $\equiv_{k+1} \subseteq \equiv_k$.

Доказательство. По лемме 8 из соотношения $X_1 \equiv_{k+1} X_2$ следует $X_1 \equiv_k X_2$. Но это и означает, что $\equiv_{k+1} \subseteq \equiv_k$ □

Лемма 10. Пусть $\equiv_k = \equiv_{k+1}$ для некоторого $k \ge 0$. Тогда $\equiv_{k+1} = \equiv_{k+2}$.

Доказательство. По лемме 8

$$X_1 \equiv_{k+2} X_2 \Leftrightarrow X_1 \equiv_{k+1} X_2 \ \ \forall a, b \in \Sigma$$

$$L(X_1\langle a, b\rangle)_{k+1} = L(X_2\langle a, b\rangle)_{k+1}.$$

По определению отношения k-неразличимости это значит, что

$$X_1 \equiv_{k+2} X_2 \Leftrightarrow X_1 \equiv_{k+1} X_2 \ \ \forall a, b \in \Sigma$$

$$X_1\langle a, b\rangle \equiv_{k+1} X_2\langle a, b\rangle.$$

Но по условию $\equiv_k = \equiv_{k+1}$. Следовательно,

$$X_1 \equiv_{k+2} X_2 \Leftrightarrow X_1 \equiv_k X_2 \ \ \forall a, b \in \Sigma$$

$$X_1\langle a, b\rangle \equiv_k X_2\langle a, b\rangle.$$

По лемме 8 получаем, что

$$X_1 \equiv_{k+2} X_2 \Leftrightarrow X_1 \equiv_{k+1} X_2,$$

т. е. $\equiv_{k+2} = \equiv_{k+1}$ $\square$

Заметим, что отношение $\equiv_0$ разбивает $2^{\mathcal{R}(M)}$ на два класса эквивалентности: содержащий терминальные пары и не содержащий таковых. Более формально

$$X_1 \equiv_0 X_2 \iff \exists P, Q \in V(M) : (P, P) \in X_1,\ (Q, Q) \in X_2$$

$$\vee\ \forall P \in V(M)\ (P, P) \notin X_1 \cup X_2.$$

Заметим также, что по лемме 8

$$X_1 \equiv_{k+1} X_2 \Leftrightarrow X_1 \equiv_k X_2 \ \ \forall a, b \in \Sigma$$

$$L(X_1\langle a, b\rangle)_k = L(X_2\langle a, b\rangle)_k.$$

По определению отношения k-неразличимости

$$X_1 \equiv_{k+1} X_2 \Leftrightarrow X_1 \equiv_k X_2 \ \ \forall a, b \in \Sigma$$

$$X_1\langle a, b\rangle \equiv_k X_2\langle a, b\rangle.$$

А это дает следующий способ вычисления отношения $\equiv_{k+1}$, если вычислено отношение $\equiv_k$.

Для каждой пары (X_1, X_2) множеств вершин, такой, что $X_1 \equiv_k X_2$, для всех a, $b \in \Sigma$ перебрать пары множеств вершин $(X_1\langle a, b\rangle, X_2\langle a, b\rangle)$. Если $X_1\langle a, b\rangle \equiv_k X_2\langle a, b\rangle$ для всех a, $b \in \Sigma$, то $X_1 \equiv_{k+1} X_2$, иначе $X_1 \not\equiv_{k+1} X_2$.

Из сделанных замечаний и лемм 9 и 10 вытекает корректность следующего алгоритма.

Алгоритм 3. Построение отношения неразличимости.

Вход. Множество $\mathcal{P}(M)$ парных дуг автомата M.

Выход. Отношение $\equiv$ неразличимости на $2^{\mathcal{R}(M)}$.

Метод. Вычислить отношение $\equiv_0$, положить k равным нулю;
вычислять $\equiv_{k+1}$ и увеличивать k на единицу, пока $\equiv_{k+1} \neq \equiv_k$;
взять в качестве отношения $\equiv$ отношение $\equiv_k$.

Утверждение 3. Алгоритм 3 заканчивает работу.

Доказательство. Конечность построения отношения $\equiv_0$ сомнений не вызывает. Как уже было замечено, отношение $\equiv_{k+1}$ строится по отношению $\equiv_k$. По лемме 9 $\equiv_{k+1} \subseteq \equiv_k$. Это значит, что либо $\equiv_{k+1} = \equiv_k$, либо $\equiv_{k+1} \subset \equiv_k$. По лемме 10 отношение $\equiv$ построено, когда $\equiv_k = \equiv_{k+1}$. Следовательно, при увеличении k на единицу и построении очередного отношения k-неразличимости либо алгоритм останавливается, либо количество пар множеств вершин, связанных отношением $\equiv_k$, уменьшается. А так как количество пар множеств вершин, связанных отношением $\equiv_0$, конечно, то алгоритм работу закончит $\square$

Сформулируем алгоритм решения проблемы эквивалентности для автоматов из $\mathcal{M}_1$.

Алгоритм 4. Проверка эквивалентности автоматов из $\mathcal{M}_1$.

Вход. Автоматы $M_i = (K_i, \Sigma, \Gamma_i, Z_{0i}, \delta_i, p_{0i}, F_i)$, $i = 1, 2$; $K_1 \cap K_2 = \emptyset$.

Выход. Если $L(M_1) = L(M_2)$, то “да", иначе “нет".

Метод. С помощью алгоритма 3 построить отношение неразличимости $\equiv$ на

$$2^{V(M_1) \cup V(M_2) \times V(M_1) \cup V(M_2)};$$

если множества пар вершин

$$\bigcup_{f_1 \in F_1} ((p_{01}, Z_{01}), (f_1, Z_{01}))$$

и

$$\bigcup_{f_2 \in F_2} ((p_{02}, Z_{02}), (f_2, Z_{02}))$$

неразличимы, то ответить “да", иначе “нет".

Утверждение 4. Алгоритм 4 ответит “да"тогда и только тогда, когда $L(M_1) = L(M_2)$.

Доказательство. Действительно, алгоритм 4 ответит “да"тогда и только тогда, когда множества

$$\bigcup_{f_1 \in F_1} ((p_{01}, Z_{01}), (f_1, Z_{01}))$$

и

$$\bigcup_{f_2 \in F_2} ((p_{02}, Z_{02}), (f_2, Z_{02}))\}$$

неразличимы, т. е. когда

$$\bigcup_{f_1 \in F_1} L((p_{01}, Z_{01}), (f_1, Z_{01})) = \bigcup_{f_2 \in F_2} L((p_{02}, Z_{02}), (f_2, Z_{02})).$$

Из определения языка пары вершин имеем

$$L(M_i) = \bigcup_{f_i \in F_1 \cup F_2} L((p_{0i}, Z_{0i}), (f_i, Z_{0i}))$$

для $i = 1, 2$. Следовательно, алгоритм 4 ответит “да"тогда и только тогда, когда $L(M_1) = L(M_2)$ $\square$

Из разрешимости проблемы эквивалентности в классе $\mathcal{M}_1$ вытекает, что для любого автомата M из этого класса можно найти множество эквивалентных ему автоматов с минимальным числом вершин следующим образом.

Пусть l — число вершин автомата M. Очевидно, существует лишь конечное (с точностью до обозначений состояний и магазинных символов) число магазинных автоматов из $\mathcal{M}_1$ над тем же алфавитом и с числом вершин, не превосходящим l. Построив множество всех таких автоматов, нужно выбрать (с помощью алгоритма 5) среди них автоматы, эквивалентные автомату M. Среди этих автоматов остается найти те, у которых число вершин минимально.

Заметим, что аналогичным образом можно минимизировать число дуг автомата.

ЛИТЕРАТУРА

1. Анисимов А.В. Синтаксические свойства рекурсивных схем программ // Кибернетика. 1977. N 3. С. 51–59.
2. Арбиб М.А., редактор. Алгебраическая теория автоматов, языков и полугрупп. М.: Статистика. 1975. 335 с.
3. Бурова Н.К., Станевичене Л.И., Станевичюс А.И.А., Шкляр П.Э. Система линейного программирования ЛП/БЭСМ6 : Гос. фонд алгоритмов и программ СССР. Рег. номер ВНТИЦ ПОО6463 от 1 июля 1983 г. // Алгоритмы и программы : Информационный бюллетень. 1983. N 5 (56). С. 23.
4. Вылиток А.А. О построении графа магазинного автомата // Вестн. Моск. унта. Сер. 15. 1996. N 3. С. 68–73.
5. Вылиток А.А. Магазинные автоматы и характеризация регулярных языков. Канд. дисс. М., МГУ. 1998.
6. Вылиток А.А., Станевичене Л.И., Чернцов И.В. Dграфы в теории магазинных автоматов. М., 1996. 64 с. Деп. в ВИНИТИ РАН 23.12.96, N 3730В96.
7. Гладкий А.В. Некоторые алгоритмические проблемы для КСграмматик // Алгебра и логика. 1965(а). Т. 4. N 1. С. 3–13.
8. Гладкий А.В. Алгоритмическая нераспознаваемость существенной неопределенности КСязыков // Алгебра и логика. 1965(б). Т. 4. N 4. С. 53–64.
9. Гладкий А.В. Формальные грамматики и языки. М., 1973.
10. Гончарова Л.И. Система СНОБОЛА–БЭСМ6 // ЖВМ и МФ. 1971. Т. 11. N 6. С. 1544–1552.
11. Гончарова Л.И. Система для обработки строк СНОБОЛА–БЭСМ6. Канд. дисс. М., ВЦ АН СССР. 1972.
12. Гончарова Л.И. Приоритетный анализ и контекстные условия // ЖВМ и МФ. 1975. Т. 15. N 3. С. 719–727.
13. Гончарова Л.И., Станевичюс А.И.А. ЛПязык // Алгоритмы и алгоритмические языки. 1971. Вып. 5. С. 94–104.
14. Диковский А.Я. Густота — мера сложности вывода в контекстноєвободной грамматике // Проблемы передачи информации. 1972(а). Т. 8. Вып. 2. С. 92–105.
15. Диковский А.Я. Густота дерева вывода и активная емкость грамматики // Проблемы передачи инфорации. 1972(б). Т. 8. Вып. 4.
16. Ершов А.П. Программирующая программа для быстродействующей электронной счетной машины. М., 1958.
17. Зубенко В.В. О конечно возвратных автоматах // Докл. АН УССР. 1978. Сер. А. N 3. С. 838–841.
18. Игнатов В.В. Расщепление грамматики и построение LR(1)анализатора // Тез. докл. 3 Всес. конф. по автоматиз. производ. систем программир. Таллин. 1986. С. 23–25.
19. Игнатов В.В. Об одном методе построения LR(1)анализатора // Вестн. Моск. унта. Сер. 15. 1987(а). N 1. С. 55–61.
20. Игнатов В.В. Построение LR(1)анализатора методом расщепления грамматики. Канд. дисс. М., МГУ. 1987(б).

21. Изимбетов Е.Т. Разработка программного обеспечения для анализа линейных производственных моделей. Канд. дисс. М., ВЦ АН СССР. 1988.
22. Кузнецова М.А., Ожиганов К.В., Сагинтаева С.Н., Станевичене Л.И. Преобразования магазинных автоматов. М., 1990. 33 с. Деп. в ВИНИТИ 26.11.90, N 5905В90.
23. Летичевский А.А. Об отношениях, представимых в pushdown автоматах // Кибернетика. 1969. N 1. С. 1–9.
24. Мартыненко Б.К. Синтаксически управляемая обработка данных. Докт. дисс. С.Петербургский гос. унт. С.Петербург. 1998.
25. Мейтус В.Ю. Детерминированные магазинные автоматы. Проблема эквивалентности. 1988. 43 с. Деп. в ВИНИТИ 03.06.88, N 4426В88.
26. Мейтус В.Ю. Проблема эквивалентности строгих детерминированных магазинных автоматов, действующих в реальном времени // Кибернетика. 1989. N 5. С. 14–25.
27. Мейтус В.Ю. Разрешимость проблемы эквивалентности детерминированных магазинных автоматов // Кибернетика и системный анаализ. 1992. N 5. С. 20–45. (Английский перевод: Meitus V. Decidability of the equivalence problem for deterministic pushdownautomata // Cybernetics and System Analysis. 1993. V. 28. P. 672–690.)
28. Мельников Б.Ф. Об одном методе исследования языков и ωязыков специального вида. М., 1989. 65 с. Деп. в ВИНИТИ 19.07.89, N 4774В89.
29. Мельников Б.Ф. Метод исследования бесконечных итерациий слов. Канд. дисс. М., МГУ. 1990.
30. Миронова И.В. Языковые и программные средства постановки задач в системах линейного программирования. Канд. дисс. М., ВЦ РАН. 1992.
31. Миронова И.В., Станевичене Л.И. Применение приоритетных C(1,1)-грамматик для описания детерминированных языков // ЖВМ и МФ. 1982. Т. 22. N 5. С. 1227–1236.
32. Непомнящая А.Ш. Проблема эквивалентности для обобщения LL(k)-грамматик // Программирование. 1981. N 5. С. 3–10.
33. Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы: Теория и практика. М.: Мир. 1980. 476 с.
34. Романовский В.Ю. Проблема эквивалентности для строгих детерминированных МПавтоматов, действующих в реальное время // Кибернетика. 1980. N 5. С. 49–59.
35. Романовский В.Ю. Проблема эквивалентности детерминированных МП-автоматов, действующих в реальное время // Кибернетика. 1986. N 2. С. 13–23.
36. Станевичене Л.И. Об одном алгоритме построения ограниченноконтекстных анализаторов // ЖВМ и МФ. 1976. Т. 16. N 5. С. 1283–1292.
37. Станевичене Л.И. Синтаксический анализ формальных языков. М., 1978.
38. Станевичене Л.И. Лекции по LR(k)анализу. М., 1983.
39. Станевичене Л.И. Вопросы теории синтаксического анализа формальных языков. М., 1984.
40. Станевичене Л.И. Средства исследования бесконтекстных языков. М., 1987(a).

41. Станевичене Л.И. К проблеме пустоты пересечения детерминированных языков // Кибернетика. 1987(б). N 6. С. 101.
42. Станевичене Л.И. Об одном средстве исследования бесконтекстных языков // Кибернетика. 1989. N 4. С. 135–136.
43. Станевичене Л.И. Графы магазинных автоматов // Учредительная конф. Российской ассоциации "Женщиныматематики". Тез. докл. М., 1993(а). С. 50.
44. Станевичене Л.И. К вопросу имитации эквивалентных детерминированных магазинных автоматов // Сб. научн. тр. Российской ассоциации "Женщиныматематики". Вып. 1. Нижний Новгород : Издво Нижегородского университета, 1993(б). С.112–117.
45. Станевичене Л.И. Одна неразрешимая алгоритмическая проблема для детерминированных магазинных автоматов // Докл. АН. 1995. Т. 342, N 6. С. 744–746.
46. Станевичене Л.И. Одна неразрешимая алгоритмическая проблема // Известия вузов. Математика. 1996(а). N 6(409)
47. Станевичене Л.И. Ядро КСграмматики и синтаксический анализ // Программирование. 1996(б). N 5. С. 19–29.
48. Станевичене Л.И. О некоторых определениях класса КСязыков // Программирование. 1999. N 5. С. 15–25.
49. Станевичене Л.И., Вылиток А.А. Один важный подкласс алгебраических языков // Третья междунар. конф. по алгебре: Тез. докл. Красноярск, 1993. С. 315–316.
50. Станевичене Л.И., Мельников Б.Ф. Об одном расширении класса минимальных линейных языков. М., 1988. 22 с. Деп. в ВИНИТИ 07.06.88, N 4500В88.
51. Станевичене Л.И., Чобан В.Б. Об одном подклассе детерминированных магазинных автоматов // Междунар. конгресс ассоциации "Женщиныматематики". Аннот. докл. М., 1994. С. 42.
52. Чернцов И.В. Одна неразрешимая алгоритмическая проблема для детерминированных магазинных автоматов, действующих в реальное время // Вестн. Моск. унта. Сер. 15. 1995. N 3. С. 71–74.
53. Шумей А.С., Зонис В.С. О синтаксическом анализе по однозначным грамматикам // Программирование. 1975. N 3. С. 22–29.
54. Яффе В.А. О двух классах КСязыков с разрешимой проблемой эквивалентности // Кибернетика. 1974. N 2. С. 89–93.
55. Aho A.V., Ullman J.D. The theory of parsing, translation and compiling. V. 1 : Parsing. N.J. 1972. (Русский перевод: Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. М. 1978. Т. 1.)
56. Aho A.V., Ullman J.D. The theory of parsing, translation and compiling. V. 2 : Compiling. N.J. 1973. (Русский перевод: Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. М. 1978. Т. 2.)
57. Barnett M.P., Futrelle R.P. Syntactic analysis by digital computer // Comm. ACM. 1962. V. 5. N 10. P. 515–526.
58. BarHillel Y., Perles M., Shamir E. On formal properties of simple phrase structure grammars // Z. Phonetik, Sprachwiss. Kommunikationsforsch. 1961. B. 14. N 2. S. 143–172.

59. Burks A.W., Warren D.W., Wright J.B. An analysis of a logimathcal machine using parenthesisfree notation // Mathematimathcal Tables and Other Aids to Computation. 1954. V. 8. P. 53–57.
60. Chomsky N. Three models for the description of Language // IEEE Trans. Inform. Theory. 1956. V. 2. N 3. P. 113–124. (Русский перевод: Хомский Н. Три модели для описания языка // Кибернетический сб. 1961. Вып. 2. С. 237–266.)
61. Chomsky N. Syntactic structures. The Hague. 1957. (Русский перевод: Хомский Н. Синтаксические структуры // Сб. "Новое в лингвистике". 1962. Вып. 11. С. 412–527.)
62. Chomsky N. On certain formal properties of grammars // Inform. and Control. 1959(a). V. 2. N 2. P. 137–167. (Русский перевод: Хомский Н. О некоторых формальных свойствах грамматик // Кибернетический сб. 1962. Вып. 5. С. 279–311.)
63. Chomsky N. A note on phrase structure grammars // Inform. and Control. 1959(b). V. 2. N 4. P. 393–395. (Русский перевод: Хомский Н. Заметка о грамматиках непосредственно составляющих // Кибернетический сб. 1962. Вып. 5. С. 312–316.)
64. Chomsky N. Contextfree grammars and pushdown storage // Quarterly Progress Report. M.I.T. Res. Lab. Electron. 1962. N 65. P. 187–194.
65. Chomsky N. Formal properties of grammars // Handbook of Mathematimathcal Psychology. 1963. V. 2. (Русский перевод: Хомский Н. Формальные свойства грамматик // Кибернетический сб. Н. С. 1966. Вып. 2. С. 121–230.)
66. Chomsky N., Schutzenberger M.P. The algebraic theory of contextfree languages // In "Computer Programming and Formal Systems". Amsterdam. 1963. (Русский перевод: Хомский Н., Шютценберже М.П. Алгебраическая теория контекстноєвободных языков // Кибернетический сб. Н. С. 1966. Вып. 3. С. 195–242.
67. Ciesinger J. Generating error recovery in a compiler generating system // InformatikFachberichte. 1979. N 1. S. 185–193.
68. DeRemer F.L. Practimathcal translators for LR(k) languages. Ph. D. Theses. M.I.T. 1969.
69. DeRemer F.L. Simple LR(k) grammars // Comm. ACM. 1971. V. 14. N 7. P. 453–460.
70. Earley J. An efficient contextfree parsing algorithm. Ph. D. Thesis. Carnegie-Melton Univ. 1968.
71. Ehrenfeucht A., Hoogeboom H.J., Rozenberg G. Coordinated pair systems // Inf. Théor. et Appl. 1986. V. 20. N 4. P. 405–439.
72. Eilenberg S. Automata, languages, and machines. N. Y.; L.: Acad. Press, 1974.
73. Elgot C.C., Mezei J.E. On relations defined by generalized finite automata // IBM J. Res. Devel. 1965. V. 9. P. 47–68.

74. Evey R.J. Applications of pushdownstore machines // Proc. AFIPS Fall Joint Computer Conference. 1963. V. 24. P. 215–227.
75. Feldman J.A., Gries D. Translator writing systems // Comm. ACM. 1968. V. 11. N 2. P. 77–113. (Русский перевод: Фельдман Дж., Грис Д. Системы построения трансляторов // Алгоритмы и алгоритмические языки. 1971. Вып. 5. С. 105–214.)
76. Floyd R.W. Syntactic analysis and operator precedence // J. ACM. 1963. V. 10. N 3. P. 316–333.
77. Floyd R.W. New proofs and old theorems in logic and formal linguistics. Wakefield, Mass. 1964.
78. Ginsburg S. Examples of abstract machines // IRE Trans. Electron. Computers. 1962. V. EC11. P. 132–135.
79. Ginsburg S. The mathematimathcal theory of contextfree languages. N.Y. 1966. (Русский перевод: Гинзбург С. Математическая теория контекстно-свободных языков. М. 1970.)
80. Ginsburg S., Greibach S. Deterministic context free languages // IEEE Conf. Record on Switch Circuit Theory and Logimathcal Design. 1965. P. 203–220.
81. Ginsburg S., Greibach S. Deterministic context free languages // Inform. and control. 1966. V. 9. N 6. P. 620–648.
82. Ginsburg S., Ullian J.S. Ambiguity in contextfree languages // J. ACM. 1966. V.13. N 3. P. 364–368.
83. Gomozov A.L., Stanevichene L.I. A Generalization of Regular Expressions // Informatica (journal of Lithuanian Academy of Sciences). 1999. V. 10. No. 1. P. 27–44.
84. Gruska J. A Characterization of ContextFree Languages // J. Comput. System Sci. 1971. V. 5. P.353364. (Русский перевод: Груска Й. Характеристика контекстноєвободных языков // Кибернетический сб. Н. С. 1973. Вып. 10. С. 114–126.)
85. Haines L.H. Generation and recognition of formal languages. Ph. D. Thesis. M.I.T. 1965.
86. Hashiguchi K. Algorithm for Determining Relative Star Height and Star Height. // Information and Computation. 1988. V.78. No. 2. P. 124–169.
87. Hirose S., Okawa S., Yoneda M. A homomorphic characterization of recursively enumerable languages // Theoret. Comput. Sci. 1985. V. 35. P. 261–269.
88. Irons E.T. A syntax directed compiler for ALGOL 60 // Comm. ACM. 1961. V. 4. N 1. P. 51–55.
89. Kleene S.C. Representation of events in nerve nets. In "Automata Studies". Princeton. 1956. (Русский перевод: Клини С.К. Представление событий в нервных сетях // Сб. "Автоматы". 1956. С. 15–67.)
90. Knuth D. On the translation of languages from left to right // Inform. and Control. 1965. V. 8. N 6. P. 607–639. (Русский перевод: Кнут Д. О переводе (трансляции) языков слева направо // Языки и автоматы. М. 1975. С. 9–42.
91. Korenjak A.J. A practimathcal method for constructing LR(k) processors // Comm. ACM. 1969. V. 12. N 11. P. 613–623.

92. Korenjak A.J., Hopcroft J.E. Simple deterministic languages // IEEE Conf. Record of 7th Annual Symposium on Switching and Automata Theory. 1966. P. 36–46. (Русский перевод: Кореньяк А., Хопкрофт Дж. Простые детерминированные языки // Сб. "Языки и автоматы". 1975. С. 71–96.)
93. Lallement G. Semigroups and combinatorial applications. N.Y. 1979. (Русский перевод: Лаллеман Ж. Полугруппы и комбинаторные приложения. М. 1985.)
94. Lee EunYung, Choe KwangMeo. Grammar coverings of a deterministic parser with action conflicts // Inf. Process. Lett. 1994. V. 51. N 2. P. 85–92.
95. Lévy J.P. Automatic correction of syntax errors in programming languages // Acta Informatica. 1975. N 4. P.271–292.
96. McWhirter I.P. Substitution expressions // J. Comput. System Sci. 1971. V. 5. P. 629–637. (Русский перевод: МакВертер И.П. Подстановочные выражения // Кибернетический сб. Н. С. 1973. Вып. 10. С. 127–136.)
97. Melnikov B. The equality condition for infinite catenations of two sets of finite words // Int. J. of Found. of Comp. Sci. 1993. V. 4. N 3. P. 267–274.
98. Melnikov B.F., Vakhitova A.A. Some more on the finite automata // Korean Journal of Computational and Applied Mathematics. 1998. V. 5. No. 3. P. 495–505.
99. Nepomnjashchaja A.Sh. Decidability of the equivalence problem for synchronous deterministic pushdown automata // Lect. Notes Comput. Sci. 1984. V. 176. P. 425–432.
100. Newell A., Shaw J.C. Programming the logic theory machine // Proc. Western Joint Computer Conf. 1957. P. 230–240.
101. Oettinger A. Automatic syntactic analysis and the pushdown store // Structure of Language and its Mathematimathcal Concepts : Proc. 12th Symposium on Applied Mathematics. 1961. P. 104–129.
102. Ore O. Theory of graphs. 1962. (Русский перевод: Оре О. Теория графов. М. 1968.)
103. Oyamaguchi M. The equivalence problem for realtime DPDAs // J. ACM. 1987. V. 34. N 3. P. 731–760.
104. Pair C. Arbres, piles et compilation // RFTI — Chiffres. 1964. V. 7. N 3. P. 199–216.
105. Post E.L. Recursive unsolvability of a problem of Thue // J. Symb. Logic. 1947. V. 12. P. 1–11.
106. Ruohonen K. On some variants of Post's correspondence problem // Acta Inf. 1983. V. 19. N 4. P. 357–362.
107. Salomaa A. Jewels of formal language theory. Rockville. 1981. (Русский перевод: Саломаа А. Жемчужины теории формальных языков. М. 1986.)
108. Salomaa K., Wood D., Yu Sh. Pumping and pushdown machines // Inf. Théor. et Appl. 1994. V. 28. N 3–4. P. 221–232.
109. Schutzenberger M.P. On contextfree languages and pushdown automata // Inform. and Control. 1963. V. 6. N 3. P. 246–264.
110. Sénizergues G. The equivalence problem for deterministic pushdown automata is decidable // Lecture Notes in Computer Science. 1997. V. 1256. P. 671–681.

111. Shankar P., Adiga B.S. A graph-based regularity test for deterministic context-free languages // Theoretimathcal Computer Science. 1991. V. 88. P. 117–125.
112. Shankar P., Adiga B.S. A graph-based regularity test for deterministic context-free languages // Theoretimathcal Computer Science. 1992. V. 95. P. 339–340.
113. Stanevichene L.I. A new method for the description of deterministic languages // Proc. of the 14th International Symp. "INFORMATICA". Yugoslavia, 1979. P. 113–116
114. Stanevichene L.I. On an approach in context-free language theory // Proc. of the Second International Conf. "Current Problems of Fundamental Sciences". V. 6. M. 1994. P. 321–324.
115. Stanevichene L.I. On the Equivalence of Pushdown Automaton Vertices // Third International Conference of Women-Mathematicians. Abstracts. Voronezh, 1995. P. 70.
116. Stanevichene L.I. DG-raphs in Context-Free Language Theory // Informatica (journal of Lithuanian Academy of Sciences). 1997. V. 8. No. 1. P. 13–26.
117. Stanevichene L.I., Choban V.B. On a Subclass of Deterministic Pushdown Automata // Proc. of Intern. Congr. of the Association "Women-Mathematicians". Moscow–Puschino, May 30–June 3, 1994. N.Novgorod, 1994. Issue 3. P. 31–35.
118. Stanevichene L.I., Vylitok A.A. An algorithm for transformation of finite automata to regular expressions // Informatica. 2000. V. 11. No. 1. P. 49–54.
119. Stearns R.E. A regularity test for pushdown machines // Inform. and Control. 1967. V. 11. N 3. P. 323–340. (Русский перевод: Стирнз Р. Проверка регулярности для магазинных автоматов // Кибернетический сб. Н. С. 1971. Вып. 8. С. 117–139.)
120. Sudkamp Th.A. Languages and machines — An introduction to the theory of computer science, Second edition. Addison-Wesley, Reading, Mass. 1997.
121. Tomita E., Seino K. The extended equivalence problem for a class of non-real-time deterministic pushdown automata // Acta Informatica. 1995. V. 32. P. 395–413.
122. Valiant L.G. Decision procedures for families of deterministic pushdown automata // Univ. of Warwick Computer Centre Report. 1973. N 7.
123. Valiant L.G. The equivalence problem for deterministic finite turn pushdown automata // Inform. and Control. 1974. N 2. P. 123–133.
124. Valiant L.G. Regularity and related problems for deterministic pushdown automata // J. Assoc. Comput. Mach. 1975. V. 22. P. 1–10.
125. Walters D.A. Deterministic contextsensitive languages // Inform. and Control. 1970. V. 17. N 1. P.14–61.
126. Wirth N. The programming language PASCAL // Acta Informatica. 1971. 1.1. P. 35–63.

127. Wirth N., Weber H. EULER — a generalization of ALGOL and its formal definition : Parts 1 and 2 // Comm. ACM. V. 9. N 1. P. 13–23. V. 9. N 2. P. 89–99.
128. Yair I., Amiram Y. New families of non real time DPDA's and their decidability results // Theor. Comput. Sci. 1984. V. 34. N 3. P. 255–274.
129. Yntema M.K. Inclusion Relations among Families of Context-Free Languages // Information and Control. 1967. V. 10. P.572–597.

СЛОВАРЬ ТЕРМИНОВ

алфавит (alphabet)
_ входной (input)
_ магазинный (pushdown)
биребро (biarc)
вершина (vertex)
_ бесполезная (useless)
_ дуги Dграфа
_ _ конечная (terminal)
_ _ начальная (initial)
_ Dграфа
_ _ входная (input)
_ _ заключительная (final)
_ дуги магазинного автомата
_ магазинного автомата
_ маршрута магазинного автомата
_ _ промежуточная (intermediate)
вывод реализует соотношение (a derivation realizes the correlation) $x \Rightarrow^* y$
вычеркивание (deletion)
вычисление (computation)
глубина (depth)
_ структуры (of a structure)
_ цепочки Dязыка (of a Dword)
гнездо (nest)
_ простое (simple)
граф
_ магазинного автомата (graph of a pushdown acceptor)
_ _ обобщенный (extended)
_ сопряжений (team graph)
длина родословной маршрута (length of a genealogy of a computing sequence)
дуга (edge)
_ бесполезная (useless)
_ каркасная (skeleton)
_ магазинного автомата
_ соединимая (joined)
_ стирающая (deleting)
_ Dграфа
заряд (charge)

_ вычисления
_ команды
_ маршрута
история развития (growth history)
_ маршрута
_ стуктуры
итерация (iteration)
_ простая (simple)
_ сложная (complicated)
канон (canon)
_ грамматики
_ Dғрафа
команда магазинного автомата (command of a pushdown acceptor)
_ накапливающая (charging)
_ нейтральная (neutral)
_ стирающая (deleting)
конфигурация
_ магазинного автомата (configuration of a pushdown acceptor)
_ _ заключительная (final)
_ _ зацикливающая (looping)
_ _ начальная (initial)
_ Dғрафа
конфликт ролей (conflict of roles)
магазинный автомат (pushdown acceptor)
_ действующий в реальное время (realtime)
_ детерминированный (deterministic)
_ наполняющий магазин в реальное время (АНРВ) (realtimeeharging)
_ сингулярный (singular)
_ совершенный (perfect)
маркер дна (магазина) (start pushdown symbol)
маршрут (computing sequence)
_ Dғрафа
_ _ накапливающий (charging)
_ _ нейтральный (neutral)
_ _ образующий (вместе с некоторым другим) гнездо (forming a nest)
_ _ открывающий (left)
_ _ парный (некоторому другому) (matching)
_ _ полный (heading)
_ _ производный (derived)
_ _ стирающий (deleting)
_ _ элементарный (elementary)
_ _ успешный (successful)
_ _ читающий (reading or labelled)
_ магазинного автомата
моноребро (monoarc)
объединимые прогнозы (unitable prognoses)
основа вычисления (base of a computation)