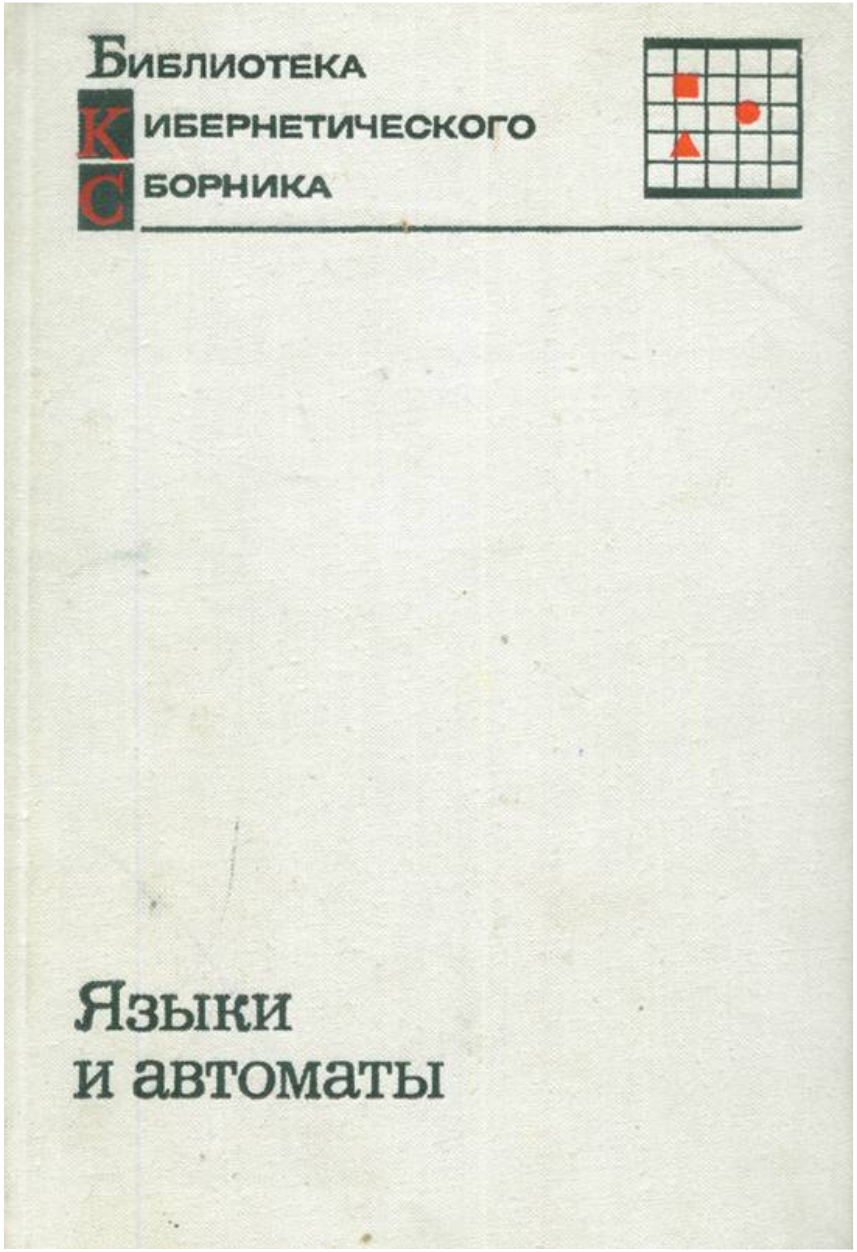


БИБЛИОТЕКА
«КИБЕРНЕТИЧЕСКОГО
СБОРНИКА»

ЯЗЫКИ И АВТОМАТЫ

Сборник переводов

Под редакцией

А. Н. МАСЛОВА и Э. Д. СТОЦКОГО

ИЗДАТЕЛЬСТВО «МИР» Москва 1975

УДК 510 : 164

Книга входит в известную серию «Библиотека «Кибернетического сборника» и содержит переводы наиболее важных статей последних лет по теории языков программирования. Эта тематика ранее затрагивалась в «Кибернетических сборниках» и сборниках переводов «Проблемы математической логики» («Мир», 1970) и «Сложность вычислений и алгоритмов» («Мир», 1974).

В сборнике три раздела. Первый посвящен контекстно-свободным языкам и открывается классической статьей Д. Кнута о языках, допускающих однопроходной анализ; в других статьях рассматриваются скорость распознавания языков и алгоритмические проблемы. Во втором разделе излагаются грамматики более общего вида — индексные, контекстные и грамматики с рассеянным контекстом. Несколько основополагающих статей третьего раздела посвящены аксиоматическому описанию языков; эта тематика еще не затрагивалась в отечественной литературе.

Книга рассчитана на специалистов по математической логике, языкам программирования, теории алгоритмов и математической лингвистике. Она будет полезна студентам и аспирантам указанных специальностей.

Редакция литературы по математическим наукам

Я $\frac{20205-026}{041(01)-75}$ 26-75

ПРЕДИСЛОВИЕ РЕДАКТОРОВ ПЕРЕВОДА

Настоящий сборник посвящен алгоритмическим способам описания формальных языков и классов языков. Статьи сгруппированы в три раздела, соответствующие основным направлениям исследований.

В первом разделе «Контекстно-свободные грамматики и автоматы с магазинной памятью» рассматриваются прежде всего методы описания языков программирования с помощью контекстно-свободных грамматик. Эти методы разработаны весьма глубоко и представляют значительный практический интерес при построении трансляторов с языков программирования. Кроме того, в этом разделе рассмотрены способы распознавания контекстно-свободных языков магазинными автоматами определенного вида; при этом автоматы позволяют строить синтаксический анализ распознаваемой фразы.

Раздел открывается статьей Д. Кнута, в которой впервые были сформулированы основные задачи синтаксического анализа языков. Известно, что каждый контекстно-свободный язык распознается некоторым недетерминированным магазинным автоматом. Однако по недетерминированному автомату в общем случае нельзя построить удобную схему распознавания фраз языка. Ввиду этого важное значение имеет выделение такого подкласса в классе языков, распознаваемых детерминированными магазинными автоматами, с которым можно сопоставить простые схемы распознавания. Наибольший интерес представляют подклассы LR(k), исследуемые в первых трех статьях раздела, и подкласс простых детерминированных языков.

К работам по синтаксическому анализу контекстно-свободных языков примыкают статьи Дж. Хопкрофта и Ш. Грейбах, в которых изучаются алгоритмические свойства грамматик. В статье У. Огдена исследуется неоднозначность грамматического описания языка и приводятся примеры контекстно-свободных языков с различной степенью неоднозначности (в том числе и с неограниченной неоднозначностью).

Во втором разделе «Расширения контекстно-свободных грамматик» представлены специальные виды грамматик, построенные

на основе контекстно-свободных грамматик, но отличающиеся существенно большей порождающей силой. Индексные грамматики, предложенные А. Ахо, относятся к числу наиболее ранних обобщений такого рода. Представляется весьма вероятным, что для описания развитых языков программирования, таких, как Алгол 68, индексные грамматики окажутся значительно удобнее контекстно-свободных. Другое весьма интересное обобщение контекстно-свободных грамматик предложено Ш. Грейбах и Дж. Хопкрофтом — это так называемые грамматики с рассеянным контекстом (которые могут рассматриваться и как естественное обобщение матричных грамматик С. Абрахама), приводящие к подклассу класса контекстных языков. До сих пор не известно, совпадает ли класс языков, порождаемых грамматиками с рассеянным контекстом, с классом контекстных языков. Исследование грамматик с рассеянным контекстом тесно связано с вопросами, возникающими при изучении условных грамматик различных типов. Подробную информацию об этом читатель может найти в обзорах [5, 6]. Раздел завершается статьей К. Чулика II и Ч. Морея, в которой обсуждается понятие перевода с одного формального языка на другой.

В третьем разделе «Аксиоматически определяемые семейства языков» рассматриваются абстрактные семейства языков и их свойства. Введенное С. Гинзбургом и Ш. Грейбах (см. первую статью раздела) понятие абстрактного семейства языков привело к развитию аксиоматического подхода в теории языков. Обзор полученных в этом направлении результатов можно найти в [5, 6]. Здесь мы отметим только одно важное свойство: для каждого абстрактного семейства языков существует семейство автоматов, допускающих языки этого семейства. Особое положение занимают те абстрактные семейства языков (называемые главными), для которых можно указать язык, порождающий данное семейство. Иными словами, каждый язык из семейства можно получить из указанного языка с помощью основных операций, входящих в определение абстрактного семейства языков.

Отметим, что в сборнике сделана попытка ввести некоторое терминологическое единообразие. Для ряда понятий мы все же по традиции сохранили различные варианты терминов, но число таких понятий очень невелико. Например, мы сочли возможным одновременно использовать термины «бесконтекстный» и «контекстно-свободный».

Сборник предназначен для специалистов по математической логике, теории алгоритмов, математической лингвистике и языкам программирования. Систематическое изложение результатов по математической лингвистике можно найти в монографиях [1, 2] (знакомство с одной из них предполагается у читателя) и обзорах [5, 6].

Переводы ряда более ранних работ по теории языков и автоматов содержатся в выпусках «Кибернетического сборника» и в сборниках [3, 4]. Настоящий сборник является продолжением сборников [3, 4].

Список литературы

1. *Гладкий А. В.* Формальные грамматики и языки, «изд-во «Наука», М., 1973.
2. *Гинзбург С.* Математическая теория контекстно-свободных языков, изд-во «Мир», М., 1970.
3. «Проблемы математической логики», изд-во «Мир», М., 1970.
4. «Сложность алгоритмов и вычислений», изд-во «Мир», М., 1974.
5. *Гладкий А. В., Диковский А. Я.* Теория формальных грамматик, в сб. «Теория вероятностей. Математическая статистика. Теоретическая кибернетика», Итоги науки, т. 10, изд-во ВИНИТИ АН СССР, М., 1972.
6. *Маслов А. Н., Стоцкий Э. Д.* О некоторых классах формальных грамматик, в сб. «Теория вероятностей. Математическая статистика. Теоретическая кибернетика», Итоги науки, т. 12, изд-во ВИНИТИ АН СССР, М. (в печати).

А. Н. Маслов, Э. Д. Стоцкий

I. КОНТЕКСТНО-СВОБОДНЫЕ ГРАММАТИКИ И АВТОМАТЫ С МАГАЗИННОЙ ПАМЯТЬЮ

О ПЕРЕВОДЕ (ТРАНСЛЯЦИИ) ЯЗЫКОВ СЛЕВА НАПРАВО[1])

Дональд Кнут

1. Введение и определения

Слово «язык» здесь будет применяться для обозначения множеств буквенных цепочек, которые обычно называются *контекстно-свободными языками* или, в терминологии других авторов, *бесконтекстными языками, (простыми) порождаемыми, языками, (простыми) языками составляющих, определимыми множествами, языками Бэкуса, языками Хамского типа* 2 (или *типа* 4), *языками типа Алгол, магазинными языками* и т. д.

Эти языки вызвали к себе широкий интерес, так как они, кроме всего прочего, служат сравнительно точными моделями для естественных языков и языков программирования. В этой статье мы выделим важный класс языков, которые назовем *переводимыми (транслируемыми) слева направо;* это значит, что, читая буквы цепочки слева направо и просматривая вперед некоторое заранее ограниченное число букв, можно проводить анализ данной цепочки, не возвращаясь назад для рассмотрения предыдущего решения. Такие языки особенно важны в случае задач программирования, поскольку это условие означает, что можно построить алгоритм анализа, время работы которого пропорционально длине цепочки, подлежащей анализу.

Специальные методы трансляции языков программирования (например, хорошо известный алгоритм предшествования [4]) основаны на том факте, что рассматриваемые языки обладают простой структурой «слева направо». При рассмотрении всех языков, которые транслируются слева направо, мы в состоянии

[1]) Knuth D., On the translation of languages from left to right, *Inform. and Control,* 8 (1965), 607—639.

изучить все эти специальные методы в их наиболее общей форме и найти для данного языка «наилучший возможный» путь трансляции его слева направо. Изучение таких языков, возможно, представляет интерес для тех, кто исследует грамматический разбор, производимый человеком; вероятно, оно поможет объяснить тот факт, что некоторые английские предложения непонятны при прослушивании.

Теперь перейдем к точным определениям понятий, неформально обсуждавшихся выше. Будем использовать общеизвестные наборы букв (символов) и цепочки (слова), построенные из букв. Введем два непересекающихся множества букв: множество I *промежуточных* (*вспомогательных*) букв, или *нетерминалов*, и множество T *терминалов*. Элементы первого множества будем обозначать прописными латинскими буквами A, B, C, ..., а элементы второго множества — строчными латинскими буквами a, b, c, ... Буквами X, Y, Z мы будем обозначать и терминалы, и нетерминалы. Обозначим через S *главный* (начальный) *нетерминальный* символ; его особое значение мы объясним ниже. Цепочки символов будут обозначаться строчными греческими буквами α, β, γ, ...; буква ε будет обозначать *пустую цепочку*. Через α^n обозначим n-кратную конкатенацию (повторение) цепочки α; $\alpha^0 = \varepsilon$ и $\alpha^n = \alpha \cdot \alpha^{n-1}$.

Правило (или *продукция*) есть отношение $A \to \theta$, где $A \in I$, а θ — цепочка букв из $I \cup T$; *грамматика* $\mathfrak{G}$ — это множество правил. Мы будем писать $\varphi \to \psi$ (относительно грамматики $\mathfrak{G}$, которая обычно подразумевается), если существуют такие цепочки α, θ, ω, A, что $\varphi = \alpha A \omega$, $\psi = \alpha\theta\omega$ и $A \to \theta$ есть правило в $\mathfrak{G}$. Существенную роль играет транзитивное замыкание этого отношения: $\alpha \Rightarrow \beta$ означает, что существуют цепочки $\alpha_0, \alpha_1, \ldots, \alpha_n$ $(n > 0)$, для которых $\alpha = \alpha_0 \to \alpha_1 \to \ldots \to \alpha_n = \beta$. Заметим, что в силу данного определения не обязательно $\alpha \Rightarrow \alpha$; мы будем писать $\alpha \Rightarrow\!\!\!\!= \beta$ для обозначения того, что $\alpha = \beta$ или $\alpha \Rightarrow \beta$. Грамматика называется *зацикливающейся*, если $\alpha \Rightarrow \alpha$ для некоторого α. (Некоторые из введенных обозначений имеют более общий характер, чем это требуется для настоящей статьи, но они введены так с целью соответствия обозначениям в родственных статьях.)

Грамматикой $\mathfrak{G}$ определяется язык

$$\{\alpha \mid S \Rightarrow \alpha \text{ и } \alpha \text{ — цепочка над } T\}, \tag{1.1}$$

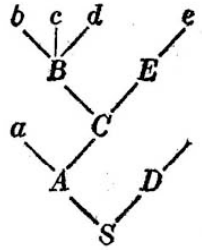
т. е. множество всех терминальных цепочек, выводимых из S при использовании правил из $\mathfrak{G}$. *Выводимой цепочкой* [1]) называется любая цепочка α, для которой $S \Rightarrow \alpha$.

[1]) Часто употребляется другой термин — *сентенциальная форма* (от английского термина sentential form). — *Прим. ред.*

Например, грамматика

$$S \to AD,\ A \to aC,\ B \to bcd,\ C \to BE,\ D \to \varepsilon,\ E \to e \tag{1.2}$$

определяет язык, состоящий из единственной цепочки *abcde*. Любую выводимую цепочку в грамматике можно представить по крайней мере одним *деревом вывода*, или диаграммой анализа; например, для цепочки *abcde* в грамматике (1.2) существует (в точности одно) дерево вывода, а именно



(1.3)

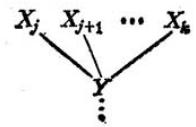
(корень дерева вывода есть S, а ветви соответствуют обычному способу применения правил). Грамматика называется *однозначной*, если каждая выводимая цепочка имеет ровно одно дерево вывода. Грамматика (1.2) очевидно однозначна, хотя возможны несколько различных *последовательностей* вывода (или просто выводов), например

$$S \to AD \to aCD \to aBED \to abcdED \to abcdeD \to abcde, \tag{1.4}$$

$$S \to AD \to A \to aC \to aBE \to aBe \to abcde. \tag{1.5}$$

Чтобы устранить различие, не имеющее для нас значения, между выводами, соответствующими одному и тому же дереву вывода, можно установить стандартный порядок, договорившись, например, что подстановка всегда осуществляется вместо самого левого (как в (1.4)) или самого правого (как в (1.5)) нетерминала.

Однако практически мы должны исходить из терминальной цепочки *abcde* и пробовать восстановить вывод, ведущий обратно к S, и это несколько меняет нашу точку зрения. Назовем *основой* (*ключом*) дерева самую левую последовательность соседних листьев, образующих законченную ветвь; в (1.3) это будет *bcd*. Другими словами, если $X_1, X_2, \ldots, X_t$ — листья дерева (где X_i при любом i есть нетерминальный или терминальный символ либо ε), мы ищем наименьшее из целых чисел k, для которых дерево имеет вид



при некоторых j и Y. Если мы рассматриваем обратное движение от $abcde$ к S, то можем представить себе, что мы начали с дерева (1.3) и удалили его основу, затем удалили основу (e) оставшегося дерева и т. д., пока не останется корень S. Этот процесс удаления основ на каждом шаге точно соответствует обращению вывода (1.5).

Читатель может легко проверить сам, что удаление основы при обращении всегда приводит к выводу, получаемому замещением на каждом шаге *самого правого* нетерминала. Это можно рассматривать как другой способ определения понятия основы. В течение процесса удаления все листья справа от основы являются терминалами, если мы начали с дерева вывода, у которого все листья — терминалы.

Нас интересует алгоритм анализа, и поэтому мы хотим научиться находить основу, когда даны только листья дерева. Занумеруем правила грамматики 1, 2, ..., s некоторым произвольным, но фиксированным образом. Пусть $\alpha = X_1 \ldots X_n \ldots X_t$ будет выводимой цепочкой, и пусть существует дерево вывода, в котором основой является цепочка $X_{r+1} \ldots X_n$, полученная применением p-го правила ($0 \leqslant r \leqslant n \leqslant t$, $1 \leqslant p \leqslant s$). Будем называть пару (n, p) *основой* цепочки α.

Грамматика называется *транслируемой слева направо с границей k* (или LR(k)-грамматикой), если выполнены следующие условия. Пусть $k \geqslant 0$ и $\dashv$ — новый символ, не входящий в $I \cup T$. Выводимая цепочка, за которой следует k символов $\dashv$, называется *k-выводимой*. Пусть $\alpha = X_1X_2 \ldots X_nX_{n+1} \ldots X_{n+k}Y_1 \ldots Y_u$ и $\beta = X_1X_2 \ldots X_nX_{n+1} \ldots X_{n+k}Z_1 \ldots Z_v$ — выводимые цепочки, в которых $u \geqslant 0$, $v \geqslant 0$ и ни один из символов $X_{n+1}, \ldots, X_{n+k}, Y_1, \ldots, Y_u, Z_1, \ldots, Z_v$ не является нетерминалом. Если (n, p) — основа цепочки α, а (m, q) — основа цепочки β, то мы требуем, чтобы было $m = n$, $p = q$. Другими словами, грамматика является LR(k) тогда и только тогда, когда всякая основа однозначно определяется цепочкой, стоящей слева от нее, и k терминалами справа от нее.

Это определение легче понять, если рассмотреть частный случай, положив $k = 1$. Предположим, что мы строим вывод типа (1.5) в обратном порядке, а рассматриваемая цепочка (за которой для удобства следует маркер $\dashv$) имеет вид $X_1 \ldots X_nX_{n+1}\alpha \dashv$, где конец $X_{n+1}\alpha \dashv$ представляет собой часть цепочки, которую мы еще не исследовали. Пусть все возможные редукции в левой части цепочки уже сделаны, так что правая граница основы должна быть в положении X_r для $r \geqslant n$. Мы хотим узнать, рассматривая следующий символ X_{n+1}, существует ли основа, для которой X_n — правая граница. Если да, то мы хотим поставить в соответствие этой основе единственную продукцию и тем самым привести (редуцировать) цепочку, а затем

повторить этот процесс; если же нет, то можно двигаться направо и читать новый символ цепочки, которую транслируем.

Описанный процесс будет совершаться тогда и только тогда, когда в грамматике выполняется следующее условие (LR(1)-условие): если $X_1X_2 \ldots X_nX_{n+1}\omega_1$ — выводимая цепочка, за которой следует $\dashv$, и все буквы подцепочки $X_{n+1}\omega_1$ — или терминалы, или $\dashv$, а эта цепочка имеет основу (n, p), то все 1-выводимые цепочки $X_1X_2 \ldots X_nX_{n+1}\omega$, для которых подцепочка $X_{n+1}\omega$ удовлетворяет упомянутому выше условию, должны иметь ту же основу (n, p). Это определение сформулировано так, чтобы иметь возможность считать равной n правую границу основы, являющейся пустой цепочкой, расположенной между X_n и X_{n+1}.

Данное выше определение LR(k)-грамматики совпадает с интуитивным понятием перевода слева направо с просматриванием вперед k букв. Предположим, что на некотором шаге перевода мы сделали все возможные редукции слева от X_n; просматривая следующие k символов $X_{n+1}, \ldots, X_{n+k}$, мы хотим узнать *независимо* от того, что следует за X_{n+k}, производить ли редукцию подцепочки $X_{r+1} \ldots X_n$, если таковая возможна. В LR(k)-грамматике этот вопрос мы в состоянии разрешить без колебаний. Если такая редукция возможна, мы произведем ее и повторим этот процесс; в противном случае мы сдвинемся на одну букву вправо.

LR(k)-грамматика очевидно однозначна, так как из ее определения следует, что у всех возможных деревьев вывода должна быть одна и та же основа, и по индукции доказывается, что у выводимой цепочки может быть лишь одно дерево вывода. Интересно отметить, что почти каждая грамматика, о которой известно, что она однозначна, есть либо LR(k)-грамматика, либо (двойственно) справа налево транслируемая грамматика, либо грамматика, транслируемая с обоих концов по направлению к середине цепочки. Таким образом, LR(k)-*условие можно рассматривать как достаточно сильный критерий однозначности.*

Как будет показано в разд. 2, при заданном k можно эффективно разрешить проблему, является ли данная грамматика LR(k)-грамматикой. Это в значительной степени связано с тем, что *все возможные конфигурации дерева, лежащие ниже его основы, можно представить в конечном автомате.*

Некоторые похожие идеи встречались раньше в литературе. Линч [9] рассмотрел специальный класс LR(1)-грамматик и доказал их однозначность. Поль [11] дал общий метод построения «слева направо» анализаторов для некоторых весьма простых LR(1)-языков. Флойд [5] и Айронс [8] независимо ввели понятие грамматик *ограниченного контекста*, обладающих таким свойством: проверяя лишь конечное число букв непосредственно

слева и справа от θ, можно узнать, редуцируется ли любая выводимая цепочка $\alpha\theta\omega$ в результате применения обращения продукции $A \to \theta$. Позднее Эйкель [3] предложил алгоритм, который строит некоторый вид магазинного распознавателя по грамматике ограниченного контекста, а Эрли [2] независимо развил сходный метод, применимый к несколько более широкому классу LR(1)-языков, но имеющий ряд серьезных дефектов. Флойд [5] также ввел более общее понятие грамматики *ограниченного правого контекста*; в нашей терминологии это LR(k)-грамматика, в которой можно эффективно узнать, является ли $X_{r+1} \ldots X_n$ основой, проверяя наряду с $X_{n+1} \ldots X_{n+k}$ только данное *конечное* число букв непосредственно слева от X_{r+1}.

В то время казалось правдоподобным, что грамматика ограниченного правого контекста — это естественная формализация интуитивного понятия грамматики, в которой возможен перевод слева направо без возвращения назад и заглядывания вперед больше, чем на данное расстояние. Однако можно показать, что конструкция Эрли дает метод анализа для некоторых грамматик, которые *не* являются грамматиками ограниченного правого контекста, хотя интуитивно они кажутся такими. Это привело к данному выше определению LR(k)-грамматики (в которой известна *вся* цепочка слева от X_{r+1}).

Естественно было бы спросить, всегда ли можно фактически анализировать цепочки в соответствии с LR(k)-грамматикой, идя слева направо. Так как существует бесконечно много цепочек $X_1 \ldots X_{n+k}$, которые должны быть использованы для анализа, нам для правильного решения этой проблемы может понадобиться «бесконечная мудрость»; определение LR(k)-грамматик утверждает только, что для каждой из этого бесконечного множества цепочек корректное решение *существует*. В разд. 2 мы покажем, что в действительности существует лишь конечное число различных возможностей.

Приведем ряд примеров для иллюстрации введенных понятий. Рассмотрим две грамматики:

$$S \to aAc, \quad A \to bAb, \quad A \to b, \tag{1.6}$$

$$S \to aAc, \quad A \to Abb, \quad A \to b. \tag{1.7}$$

Обе они однозначны и определяют один и тот же язык $\{ab^{2n+1}c\}$. Грамматика (1.6) не является LR(k) ни для какого k, так как, просматривая начало цепочки ab^n, мы не получаем никакой информации о том, можно ли заменить b на A; анализ должен ждать до тех пор, пока не будет прочитано c. С другой стороны, грамматика (1.7) есть LR(0), а фактически это язык ограниченного контекста; выводимые цепочки — $\{aAb^{2n}c\}$ и $\{ab^{2n+1}c\}$, и для анализа мы должны редуцировать подцепочку ab к aA, подцепочку Abb к A, подцепочку aAc к S. Этот пример показывает, что LR(k)-свойство является свойством *грамматики*, а не одного лишь *языка*. Различие между грамматикой и языком чрезвычайно важно, когда семантика рассматривается в той же мере, что и синтаксис.

Грамматика

$$S \to aAd, \quad S \to bAb, \quad A \to cA, \quad A \to c, \quad B \to d \tag{1.8}$$

имеет выводимые цепочки

$$\{ac^nAd\} \cup \{ac^{n+1}d\} \cup \{bc^nAB\} \cup \{bc^nAd\} \cup \{bc^{n+1}B\} \cup \{bc^{n+1}d\}.$$

В цепочке $bc^{n+1}d$ надо заменить d на B, а в цепочке $ac^{n+1}d$ такую замену делать не следует. Таким образом, решение зависит от неограниченного числа символов слева от d, так что грамматика не является грамматикой ограниченного контекста (а также и грамматикой, транслируемой справа налево). С другой стороны, очевидно, что это LR(1)-грамматика, а в действительности она еще и грамматика ограниченного правого контекста, поскольку основа сразу определяется при рассмотрении одного символа справа и двух символов слева от нее. После прочтения символа d выводимая цепочка должна будет редуцироваться либо к aAd, либо к bAd.

Грамматика

$$S \to aA, \quad S \to bB, \quad A \to cA, \quad A \to d, \quad B \to cB, \quad B \to d \tag{1.9}$$

не есть грамматика ограниченного правого контекста, так как в обеих цепочках ac^nd и bc^nd основой является d, однако это, несомненно, LR(0)-грамматика. Гораздо интереснее пример грамматики

$$S \to aAc, \quad S \to b, \quad A \to aSc, \quad A \to b. \tag{1.10}$$

Здесь терминальными цепочками будут $\{a^nbc^n\}$, и однобуквенные цепочки b должны быть редуцированы к S или A в зависимости от четности n. Это еще одна LR(0)-грамматика, не являющаяся грамматикой ограниченного правого контекста.

В разд. 3 мы приведем другие примеры и обсудим смысл и отношение введенных понятий к грамматике Алгола 60. Раздел 4 содержит доказательство того, что проблема, будет ли хотя бы при одном k данная грамматика LR(k)-грамматикой, рекурсивно неразрешима. Гинзбург и Грейбах [7] ввели понятие *детерминированных языков*; мы покажем в разд. 5, что это в точности те языки, для которых *существуют* порождающие их LR(k)-грамматики (k зависит от языка), и отсюда получим ряд интересных следствий.

2. Анализ LR(k)-грамматик

Пусть дано целое число $k \geqslant 0$. Мы опишем сейчас два способа проверки того, является ли данная грамматика $\mathfrak{G}$ LR(k)-грамматикой. Как обычно, предположим, что в $\mathfrak{G}$ нет бесполезных правил, т. е. для каждого A в I существуют такие терминальные цепочки α, β, γ, что $S \Longrightarrow \alpha A \gamma \Rightarrow \alpha\beta\gamma$. Первый способ проверки состоит в построении грамматики $\mathfrak{F}$, отражающей все возможные конфигурации из основы и k букв справа от нее. Нетерминальными символами грамматики $\mathfrak{F}$ будут $[A; \alpha]$, где $A \in I$ и α есть k-буквенная цепочка в алфавите $T \cup \{\dashv\}$, а также $[p]$, где p — номер продукции из $\mathfrak{G}$. Терминальным алфавитом грамматики $\mathfrak{F}$ пусть будет $I \cup T \cup \{\dashv\}$.

Для удобства обозначим через $H_k(\sigma)$ множество всех k-буквенных цепочек β в алфавите $T \cup \{\dashv\}$, для которых $\sigma \Longrightarrow \beta\gamma$ относительно $\mathfrak{G}$ при некотором γ; это множество [1]) всевозможных начальных подцепочек длины k в цепочках, выводимых из σ.

Пусть p-я продукция грамматики $\mathfrak{G}$ будет

$$A_p \to X_{p1} \ldots X_{pn_p}, \quad 1 \leqslant p \leqslant s, \quad n_p \geqslant 0. \tag{2.1}$$

Построим все правила вида

$$[A_p; \alpha] \to X_{p1} \ldots X_{p(j-1)}[X_{pj}; \beta], \tag{2.2}$$

где $1 \leqslant j \leqslant n_p$, X_{pj} — нетерминал, а α и β обозначают k-буквенные цепочки в алфавите $T \cup \{\dashv\}$, причем $\beta \in H_k(X_{p(j+1)} \ldots \ldots X_{pn_p}\alpha)$. Добавим еще правила

$$[A_p; \alpha] \to X_{p1} \ldots X_{pn_p}\alpha[p]. \tag{2.3}$$

Теперь легко видеть, что относительно грамматики $\mathfrak{F}$

$$[S; \dashv^k] \Rightarrow X_1 \ldots X_n X_{n+1} \ldots X_{n+k}[p] \tag{2.4}$$

тогда и только тогда, когда существует k-выводимая цепочка $X_1 \ldots X_n X_{n+1} \ldots X_{n+k} Y_1 \ldots Y_u$ с основой (n, p) и терминалами $X_{n+1}, \ldots, Y_u$. Поэтому, согласно определению, $\mathfrak{G}$ будет LR(k)-грамматикой тогда и только тогда, когда $\mathfrak{F}$ удовлетворяет следующему условию:

$$[S; \dashv^k] \Rightarrow \theta[p] \text{ и } [S; \dashv^k] \Rightarrow \theta\varphi[q] \text{ влечет } \varphi = \varepsilon \text{ и } p = q. \tag{2.5}$$

Но $\mathfrak{F}$ — *регулярная* [2]) грамматика, а для проверки условия (2.5) в регулярных грамматиках существуют хорошо известные

[1]) Для построения множеств $H_k(\sigma)$ полезно построить грамматику $\mathfrak{G}'$, эквивалентную $\mathfrak{G}$, но не содержащую правил с пустой правой частью (см. [13]). — *Прим. ред.*

[2]) См. [13]. — *Прим. перев.*

методы. (В основном они сводятся к преобразованию сначала грамматики $\mathfrak{F}$ так, чтобы все ее правила приняли вид $Q_i \to aQ_j$, а затем, если $Q_0 = [S; \dashv^k]$, к систематической заготовке списка всех пар (i, j), для которых найдется такая цепочка α, что $Q_0 \to \alpha Q_i$ и $Q_0 \Rightarrow \alpha Q_j$.)

При $k = 2$ грамматика $\mathfrak{F}$, соответствующая (1.2), такова:

$$\begin{array}{ll}
[S; \dashv\dashv] \to [A; \dashv\dashv], & [C; \dashv\dashv] \to [B; e\dashv], \\
[S; \dashv\dashv] \to A[D; \dashv\dashv], & [C; \dashv\dashv] \to B[E; \dashv\dashv], \\
[S; \dashv\dashv] \to AD\dashv\dashv[1], & [C; \dashv\dashv] \to BE\dashv\dashv[4], \\
[A; \dashv\dashv] \to a[C; \dashv\dashv], & [B; e\dashv] \to bcde\dashv[3], \\
[A; \dashv\dashv] \to aC\dashv\dashv[2], & [E; \dashv\dashv] \to e\dashv\dashv[6], \\
[D; \dashv\dashv] \to \dashv\dashv[5]. &
\end{array} \tag{2.6}$$

Конечно, не обязательно перечислять правила, которые нельзя достичь из $[S; \dashv\dashv]$. Условие (2.5) проверяется непосредственно; можно усмотреть тесную связь между (2.6) и деревом (1.3).

Наш второй способ проверки LR(k)-условия связан с первым, но он, по-видимому, естественнее и в то же время дает метод анализа грамматики $\mathfrak{G}$, если она действительно является LR(k)-грамматикой. Метод анализа усложняется при появлении в грамматике ε, когда становится необходимым произвести очень тщательный разбор следующей проблемы: вставлять ли нетерминал A, отвечающий продукции $A \to \varepsilon$? Чтобы надлежащим образом изложить соответствующее условие, определим множество $H'_k(\sigma)$ так же, как $H_k(\sigma)$, но теперь опустим все выводы, содержащие шаги вида $A\omega \to \omega$, т. е. такие, в которых нетерминал, будучи *первой буквой*, заменяется на ε. Это означает, что мы отбрасываем деревья вывода, основами которых являются крайние левые пустые цепочки. Например, в грамматике

$$S \to BC\dashv\dashv\dashv,\ B \to Ce,\ B \to \varepsilon,\ C \to D,\ C \to Dc,\ D \to \varepsilon,\ D \to d$$

мы получаем

$$H_3(S) = \{\dashv\dashv\dashv, c\dashv\dashv, ce\dashv, cec, ced, d\dashv\dashv, dce, de\dashv, dec, ded, e\dashv\dashv, ec\dashv, ed\dashv, edc\},$$

$$H'_3(S) = \{dce, de\dashv, dec, ded\}.$$

Как и раньше, мы предполагаем, что продукции грамматики $\mathfrak{G}$ записаны в виде (2.1). Изменим $\mathfrak{G}$, вводя новый нетерминал S_0, добавляя «нулевую» продукцию

$$S_0 \to S\dashv^k \tag{2.7}$$

и рассматривая S_0 как начальный (главный) нетерминал. Выводимые цепочки теперь совпадают с определенными выше k-выводимыми цепочками — в этом основное преимущество метода.

Наша конструкция основана на понятии *положения*, которое будет обозначаться через $[p, j; \alpha]$; здесь p — номер продукции, $0 \leqslant j \leqslant n_p$, α есть k-буквенная терминальная цепочка. Интуитивно: мы попадем в положение $[p, j; \alpha]$, если анализ, проведенный до этого момента, привел к цепочке $\beta X_{p1} \ldots X_{pj}$ и $\mathfrak{G}$ содержит выводимую цепочку $\beta A_p \alpha \ldots$; иными словами, мы нашли первые j букв, необходимые для того, чтобы завершить p-ю продукцию, а α — цепочка, которая может законно следовать за всей продукцией, если последняя завершится.

В каждый момент трансляции мы оказываемся в *множестве* $\mathscr{S}$ положений. Ясно, что существует лишь конечное число возможных множеств положений, хотя это число огромно. Мы надеемся, что множеств положений, которые действительно могут возникнуть в процессе трансляции, не окажется слишком много. Для каждого из этих возможных множеств положений мы дадим правило, разъясняющее, какой шаг в анализе следует совершить и в какое новое множество положений перейти.

Через весь процесс трансляции мы сохраняем *стек*, обозначаемый

$$\mathscr{S}_0 X_1 \mathscr{S}_1 X_2 \mathscr{S}_2 \ldots X_n \mathscr{S}_n \,|\, Y_1 \ldots Y_k \omega. \tag{2.8}$$

Часть слева от вертикальной линии состоит из множеств положений и букв; она представляет собой часть цепочки, которая уже транслирована (возможно, за исключением основы), и множества положений $\mathscr{S}_i$, в которых мы оказались сразу после рассмотрения $X_1 \ldots X_i$. Справа от вертикальной черты стоят k терминальных символов $Y_1 \ldots Y_k$, которые можно использовать для управления трансляцией, а за ними идет еще не просмотренная цепочка ω.

Определение процесса трансляции индуктивно. Сначала мы находимся в множестве положений $\mathscr{S}_0$, содержащем единственное положение $[0, 0; \dashv^k]$, стек слева от вертикальной черты в (2.8) содержит только $\mathscr{S}_0$, а цепочка, подлежащая анализу (за которой следует $\dashv^k$), стоит справа от вертикальной черты. Предположим, что на данном этапе трансляции содержимое стека задано выражением (2.18), а мы находимся в множестве положений $\mathscr{S} = \mathscr{S}_n$.

Шаг 1. Вычислим *замыкание* $\mathscr{S}'$ множества $\mathscr{S}$, рекурсивно определяемое как наименьшее из множеств, удовлетворяющих условию

$$\mathscr{S}' = \mathscr{S} \cup \{[q, 0; \beta] \,|\, \exists\, [p, j; \alpha] \in \mathscr{S}' \quad (j < n_p,\ X_{p(j+1)} = A_q \text{ и } \beta \in H_k(X_{p(j+2)} \ldots X_{pn_p}\alpha))\}. \tag{2.9}$$

(Таким образом, мы добавили к $\mathscr{S}$ все продукции, которые можно применить в дополнение к тем, что мы уже применили.)

Шаг 2. Построим множества k-буквенных цепочек:

$$Z = \{\beta \,|\, \exists [p, j; \alpha] \in \mathscr{S}' \,(j < n_p,\ \beta \in H'_k(X_{p(j+1)} \ldots X_{pn_p}\alpha))\}, \tag{2.10}$$

$$Z_p = \{\alpha \,|\, [p, n_p; \alpha] \in \mathscr{S}'\}, \quad 0 \leqslant p \leqslant s. \tag{2.11}$$

В Z представлены все цепочки $Y_1 \ldots Y_k$, для которых соответствующая основа *не* содержится в стеке, а в Z_p представлены все цепочки, перед которыми должна быть использована p-я продукция, чтобы редуцировать стек. Поэтому или множества $Z, Z_0, \ldots, Z_s$ *попарно не пересекаются*, или грамматика не является LR(k). Эти формулы и замечания имеют смысл даже для случая $k = 0$.

Предположим, что множества $Z, Z_0, \ldots, Z_s$ попарно не пересекаются. Тогда цепочка $Y_1 \ldots Y_k$ должна принадлежать одному из них, так как в противном случае выявлена ошибка, т. е. анализируемое слово не порождается грамматикой. Если $Y_1 \ldots Y_k$ принадлежит Z, сдвинем стек целиком влево:

$$\mathscr{S}_0 X_1 \mathscr{S}_1 \ldots \mathscr{S}_n Y_1 \,|\, Y_2 \ldots Y_k \omega.$$

Переобозначим его буквы, положив $X_{n+1} = Y_1$, $Y_1 = Y_2$, ...:

$$\mathscr{S}_0 X_1 \mathscr{S}_1 \ldots \mathscr{S}_n X_{n+1} \,|\, Y_1 \ldots Y_k \omega'.$$

Затем перейдем к шагу 3.

Если $Y_1 \ldots Y_k$ принадлежит Z_p, положим $r = n - n_p$. Теперь стек содержит цепочку $X_{r+1} \ldots X_n$, равную правой части продукции p. Заменим содержимое стека (2.8) на

$$\mathscr{S}_0 X_1 \mathscr{S}_1 \ldots X_r \mathscr{S}_r A_p \,|\, Y_1 \ldots Y_k \omega \tag{2.12}$$

и положим $n = r$, $X_{n+1} = A_p$. (Заметим, что здесь использованы очевидные соглашения об обозначениях при обращении с пустыми цепочками; имеем $0 \leqslant r \leqslant n$. Если $n_p = 0$, т. е. если правая часть продукции p пуста, длина стека при переходе от (2.8) к (2.12) *увеличивается*, в противном случае стек укорачивается.)

Шаг 3. Теперь стек имеет вид

$$\mathscr{S}_0 X_1 \mathscr{S}_1 \ldots X_n \mathscr{S}_n X_{n+1} \,|\, Y_1 \ldots Y_k \omega. \tag{2.13}$$

Вычислим $\mathscr{S}'_n$ в соответствии с (2.9) и определим новое множество

$$\mathscr{S}_{n+1} = \{[p, j+1; \alpha] \,|\, [p, j; \alpha] \in \mathscr{S}'_n \text{ и } X_{n+1} = X_{p(j+1)}\}. \tag{2.14}$$

Заметим, что $\mathscr{S}_{n+1}$ — это то множество положений, в которое мы теперь перешли; мы вставляем $\mathscr{S}_{n+1}$ в стек (2.13) сразу же слева от вертикальной черты и возвращаемся к шагу 1 с $\mathscr{S} = \mathscr{S}_{n+1}$ и с n, увеличенным на 1. Однако если $\mathscr{S}$ теперь равно $[0, 1; \dashv^k]$ и $Y_1 \ldots Y_k = \dashv^k$, то анализ выполнен.

Это завершает конструкцию метода анализа. Метод, охватывающий наиболее общий случай, неизбежно получается сложным, потому что вся относящаяся к делу информация должна сохраняться. Структура этого общего метода должна пролить свет на важные частные случаи, которые имеют место, когда LR(k)-грамматика является грамматикой более простого типа.

Мы не приводим формального доказательства того, что этот метод анализа действительно правильно работает, так как читатель легко может проверить, что на каждом шаге утверждения, которые мы сделали о множестве положений и стеке, остаются справедливыми. Построение всех возможных множеств положений, которые могут возникнуть, должно завершиться, поскольку существует лишь конечное число таких множеств. Грамматика будет LR(k), если множества Z, Z_p $(0 \leqslant p \leqslant s)$ из (2.10) и (2.11) для каждого возможного множества положений не пересекаются. Описанный только что метод анализа заканчивает работу, так как любая цепочка в языке имеет конечный вывод, а каждое исполнение шага 2 либо находит шаг в выводе, либо редуцирует длину цепочки, которая еще не была проверена.

3. Примеры

Приведем три примера приложений описанного метода к некоторым нетривиальным языкам. Сначала рассмотрим грамматику

$$S = \varepsilon,\ S \to aAbS,\ S \to bBaS,$$
$$A \to \varepsilon,\ A \to aAbA,\ B \to \varepsilon,\ B \to bBaB, \qquad (3.1)$$

терминальными цепочками которой являются *все цепочки в алфавите* $\{a, b\}$, *имеющие одинаковое число вхождений букв* a *и* b. Есть основание считать, что (3.1) — кратчайшая возможная однозначная грамматика для этого языка. Мы докажем однозначность, показав, что эта грамматика есть LR(1). Для доказательства используем первую конструкцию из разд. 2. Грамматикой $\mathfrak{F}$ будет

$[S; \dashv] \to \dashv [1]$,
$[S; \dashv] \to a[A; b]$, $[S; \dashv] \to aAb[S; \dashv]$, $[S; \dashv] \to aAbS \dashv [2]$.
$[S; \dashv] \to b[B; a]$, $[S; \dashv] \to bBa[S; \dashv]$, $[S; \dashv] \to bBaS \dashv [3]$,
$[A; b] \to b[4]$,
$[A; b] \to a[A; b]$, $[A; b] \to aAb[A; b]$, $[A; b] \to aAbAb[5]$,
$[B; a] \to a[6]$,
$[B; a] \to b[B; a]$, $[B, a] \to bBa[B; a]$ $[B; a] \to bBaBa[7]$.

Поэтому цепочками, входящими в условие (2.5), будут

$(aAb, bBa)*\dashv[1]$, $(aAb, bBa)*aAbS\dashv[2]$, $(aAb, bBa)*bBaS\dashv[3]$,
$(aAb, bBa)*a(a, aAb)*b[4]$, $(aAb, bBA)*a(a, aAb)*aAbAb[5]$,
$(aAb, bBa)*b(b, bBa)*a[6]$, $(aAb, bBa)*b(b, bBa)*bBaBa[7]$.

Здесь $(\alpha, \beta)*$ обозначает множество всех цепочек, которые можно образовать из α и β последовательной конкатенацией [1]). Очевидно, что условие (2.5) выполнено.

Приведем еще один очень интересный пример. Рассмотрим сначала множество всех цепочек, полученных в результате *полной расстановки скобок* в алгебраических выражениях, включающих букву a и знак бинарной операции $+$:

$$S \to a, \quad S \to (S + S). \qquad (3.2)$$

В этой грамматике a, $+$, (и) являются терминалами. Выберем произвольную цепочку, выводимую в этой грамматике, и преобразуем ее следующим образом:

(i) Все знаки $+$ стираются.
(ii) Крайние левые и крайние правые скобки стираются.
(iii) Все левые и правые скобки заменяются буквой b.

Вопрос. Можно ли после всех этих преобразований восстановить исходную цепочку? Ответ оказывается неожиданным — да. Нетрудно видеть, что этот вопрос эквивалентен следующему: возможен ли однозначный анализ любой терминальной цепочки в грамматике

Номер продукции	Продукция	Номер продукции	Продукция
0	$S \to B \dashv$		
1	$B \to a$	2	$B \to LR$
3	$L \to a$	4	$L \to LNb$
5	$R \to a$	6	$R \to bNR$
7	$N \to a$	8	$N \to bNNb$

(3.3)

Здесь B, L, R, N обозначают множества цепочек, полученных из (3.2) чередованием преобразований (i) и (iii) с удалением скобок с *обоих* концов, с *левого* конца, с *правого* конца и со *скобками на обоих концах* соответственно.

Сразу неясно, что грамматика (3.3) однозначна, так же как сразу неясно, как построить для нее эффективный алгоритм анализа. Вторая конструкция из разд. 2 показывает, однако, что

[1]) Фактически $(\alpha, \beta)*$ есть итерация объединения α и β; обычно она обозначается через $(\alpha \cup \beta)^*$. — *Прим. ред.*

эта грамматика является LR(1)-грамматикой, и это дает нам метод анализа. Это отражено подробно в табл. 1.

Таблица 1

Метод анализа для грамматики (3.3)

Множество положений $\mathscr{S}$	Дополнительные положения в $\mathscr{S}'$	Если Y_1 есть	то	Если X_{n+1} есть	то перейти к
$00\,\dashv$	$10\,\dashv\ 20\,\dashv\ 30ab\ 40ab$	a	сдвинуть	B	$01\,\dashv$
				a	$11\,\dashv\ 31ab$
				L	$21\,\dashv\ 41ab$
$01\,\dashv$		$\dashv$	остановиться		
$11\,\dashv\ 31ab$		$\dashv$	редуцировать 1		
		$a,\ b$	редуцировать 3		
$21\,\dashv\ 41ab$	$50\,\dashv\ 60\,\dashv\ 70b\ 80b$	$a,\ b$	сдвинуть	R	$22\,\dashv$
				N	$42ab$
				a	$51\,\dashv\ 71b$
				b	$61\,\dashv\ 81b$
$22\,\dashv$		$\dashv$	редуцировать 2		
$42ab$		b	сдвинуть	b	$43ab$
$51\,\dashv\ 71ab$		$\dashv$	редуцировать 5		
		$a,\ b$	редуцировать 7		
$61\,\dashv\ 81ab$	$70ab\ 80ab$	$a,\ b$	сдвинуть	N	$61\,\dashv\ 82ab$
				a	$71ab$
				b	$81ab$
$43ab$		$a,\ b$	редуцировать 4		
$62\,\dashv\ 82ab$	$50\,\dashv\ 60\,\dashv\ 70b\ 80b$	$a,\ b$	сдвинуть	R	$63\,\dashv$
				N	$84ab$
				a	$51\,\dashv\ 71b$
				b	$61\,\dashv\ 81b$
$63\,\dashv$		$\dashv$	редуцировать 6		
$84ab$		$a,\ b$	редуцировать 8		

В табл. 1 символ $21\,\dashv$ обозначает положение $[2, 1; \dashv]$, а $41\,ab$ — два положения $[4, 1; a]$ и $[4, 1; b]$. «Сдвинуть» означает «произвести сдвигающую влево операцию», упомянутую на шаге 2; «редуцировать p» означает «произвести преобразование (2.12) с правилом p». Первые строки табл. 1 образуются так: для данного начального множества положений $\mathscr{S} = \{00\dashv\}$ надо построить $\mathscr{S}'$ в соответствии с (2.9). Так как $X_{01} = B$ и $X_{02} = \dashv$, то в $\mathscr{S}'$ следует включить $10\,\dashv$ и $20\,\dashv$. Так как $X_{21} = L$ и $X_{22} = R$, то в $\mathscr{S}'$ надо включить $30\,ab$ и $40\,ab$ (где a, b — возможные начальные буквы для $R\,\dashv$). Так как $X_{41} = L$ и $X_{42} = N$, то аналогично в $\mathscr{S}'$ надо включить $30\,ab$ и $40\,ab$, но они уже включены, и потому множество $\mathscr{S}'$ полностью определено. Теперь $Z = \{a\}$, так что на шаге 2 остается единственная возможность: убедиться в том, что $Y_1 = a$, и сдвинуть. Шаг 3 интереснее: если мы когда-нибудь придем к шагу 3 с $\mathscr{S}_n = \mathscr{S}$ (это охватывает более поздние случаи, когда редукция (2.12) уже произведена), то для X_{n+1} будут три возможности. Они определяются семью положениями из $\mathscr{S}'$, и правый столбец является просто результатом применения равенства (2.14).

В табл. 1 использовано следующее важное сокращение. Хотя переход в множество положений $51\,\dashv 71\,b$ возможен, в таблицу оно не вошло; это связано с тем, что $51\,\dashv 71\,b$ *содержится в* $51\,\dashv 71\,ab$. Процедура для данного множества положений должна совпадать с процедурой для любого его подмножества. (Из-за этого на шаге 2 можно обнаружить меньше ошибок, но мы вскоре оправдаем это уменьшение.) Часто можно взять объединение нескольких множеств положений, для которых анализ не приводит к конфликту. Это значительно сократит алгоритм анализа, порожденный конструкцией из разд. 2.

Если на шаге 2 встретилась только одна возможность, то нет необходимости проверять цепочку $Y_1 \ldots Y_k$; например, в табл. 1 (первая строка) не обязательно проверять, что $Y_1 = a$. Обнаруживать ошибку не обязательно до тех пор, пока не появляется необходимость сдвинуть $Y_1 = \dashv$ влево от вертикальной черты. В этот момент стек будет содержать $\mathscr{S}_0 S \mathscr{S}_1 |\ \dashv^k$ тогда и только тогда, когда входная цепочка правильно построена; в самом деле, правильно построенные цепочки будут проанализированы, а (по определению!) неправильно построенные цепочки нельзя редуцировать к $S\,\dashv^k$ применением обращений продукций. Следовательно, обнаружение любого числа или даже всех ошибок можно отложить до конца. (Если $k = 0$, то символ $\dashv$ должен появиться справа, чтобы выявить эту задержанную ошибку.)

Вряд ли можно написать статью об анализе без традиционного примера арифметических выражений. Рассмотрим типичную грамматику

Номер продукции	Продукция	Номер продукции	Продукция
0	$S \to E\,\dashv$	4	$T \to P$
1	$E \to -\,T$	5	$T \to T * P$
2	$E \to T$	6	$P \to a$
3	$E \to E - T$	7	$P \to (E)$

(3.4)

Терминальным алфавитом этой грамматики служит множество $\{a, —, *, (,), \dashv\}$; например, языку, порождаемому этой грамматикой, принадлежит цепочка $a — (— a * a — a) \dashv$. В табл. 2 показан анализ, использующий наш метод. В строке 10 цифры 4, 5, 6, фигурирующие в столбце X_{n+1}, означают, что правила 4, 5 и 6 применяются также и к этому множеству положений. Такая «факторизация» правил дает еще один способ упрощения анализа, использующего наш метод. Читатель, без сомнения, увидит и другие возможности упрощения табл. 2.

С помощью нашей конструкции можно определить в точности, какая информация об анализируемой цепочке известна в каждый данный момент. Располагая этими сведениями, можно узнать, какая доля информации в действительности не существенна (т. е. какова избыточность), и тем самым определить «наилучший возможный» (в некотором смысле) метод анализа для грамматик. К упрощениям этого типа относятся и упоминаемые выше два упрощения (проверка задержанной ошибки и взятие объединений взаимозаменяемых множеств положений). Для дальнейшего анализа этой проблемы необходимо дополнительное исследование.

Во многих случаях запоминать множества положений $\mathscr{P}_i$ на стеке не обязательно, так как положения из $\mathscr{P}_r$, используемые на шаге 2, часто можно определить из рассмотрения некоторых X_j у вершины стека. Например, в случае грамматики ограниченного правого контекста, определенной в разд. 1, это всегда можно сделать. Обе грамматики (3.3) и (3.4) являются грамматиками ограниченного контекста.

Из табл. 1 видно, как можно восстановить необходимую информацию о множестве положений, не запоминая ее в стеке: достаточно рассмотреть те множества положений, у которых в столбце X_{n+1} есть по крайней мере один нетерминал, так как другие множества положений не нужны анализатору. Теперь сразу же ясно из табл. 1, что $\{0\,0\ \dashv\}$ всегда является дном стека, $\{21\ \dashv, 41\ ab\}$ всегда справа от L, $\{61\ \dashv, 81\ ab\}$ всегда справа от b, а $\{62\ \dashv, 82\ ab\}$ всегда справа от N.

Грамматика (3.4) связана с определением арифметических выражений в Алголе 60, и естественно задаться вопросом, является ли Алгол 60 LR(k)-языком. Ответ несколько затруднителен, так как Алгол 60 определен [10] не только в терминах продукций: существуют «соглашения о комментариях» и эпизодические неформальные объяснения. Эта грамматика не может быть LR(k)-грамматикой, так как у нее есть некоторые синтаксические неоднозначности, например имеется правило ⟨отрытая строка⟩ → ⟨открытая строка⟩⟨открытая строка⟩, которое всегда дает неоднозначность. Неоднозначности другого типа возникают, когда ⟨идентификатор⟩ употребляется как ⟨фактический пара-

Таблица 2

Метод анализа для грамматики (3.4)

Множество положений $\mathscr{P}$	Дополнительные положения в $\mathscr{P}$	Y_1	Действия на шаге 2	Номер правила	X_{n+1}	Перейти к
00 71 ⊣) — •	10 ⊣)—20 ⊣)—30 ⊣)	— (*a*	сдвинуть	1	*E*	01 ⊣72 ⊣)— • 31 ⊣)—
	40 ⊣) — • 50 ⊣) — •			2	—	11 ⊣) —
	60 ⊣) — • 70 ⊣) — •			3	*T*	21 ⊣) — 51 ⊣ — •
				4	*P*	41 ⊣) — •
				5	*a*	61 ⊣) — •
				6	(	71 ⊣) — •
01 72 ⊣) — • 31 ⊣) —		⊣	остановить	7	)	73 ⊣) — •
		) —	сдвинуть	8	—	32 ⊣) —
11 ⊣) —	40 ⊣) — • 50 ⊣) — •			9	*T*	12 ⊣) — 51 ⊣) — •
	60 ⊣) — • 70 ⊣) — •				4, 5, 6	
21 ⊣) — 51 ⊣) — •		*	сдвинуть	10	*	52 ⊣) — •
		⊣) —	редуцировать 2			
32 ⊣) —	40 ⊣) — • 50 ⊣) — •			11	*T*	33 ⊣) — 51 ⊣) — •
	60 ⊣) — • 70 ⊣) — •				4, 5, 6	
12 ⊣) — 51 ⊣) — •		*	сдвинуть	12	*	52 ⊣) — •
		⊣) —	редуцировать 1			
52 ⊣) — •	60 ⊣) — • 70 ⊣) — •	(*a*	сдвинуть	13	*P*	53 ⊣) — •
					5, 6	
33 ⊣) — 51 ⊣) — •		*	сдвинуть	14	*	52 ⊣) — •
		⊣) —	редуцировать 3.			
pn_6X		*X*	редуцировать *p*			

метр⟩. Это бывает в следующих 8 случаях:

⟨фактический параметр⟩ → ⟨идентификатор массива⟩ → → ⟨идентификатор⟩,

⟨фактический параметр⟩ → ⟨идентификатор переключателя⟩ → → ⟨идентификатор⟩,

⟨фактический параметр⟩ → ⟨идентификатор процедуры⟩ → → ⟨идентификатор⟩,

⟨фактический параметр⟩ → ⟨выражение⟩ → → ⟨именующее выражение⟩ ⇒ ⇒ ⟨идентификатор⟩,

⟨фактический параметр⟩ → ⟨выражение⟩ → → ⟨логическое выражение⟩ ⇒ ⇒ ⟨переменная⟩ ⇒ ⟨идентификатор⟩,

⟨фактический параметр⟩ → ⟨выражение⟩ → ⟨логическое выражение⟩ ⇒ ⟨указатель функции⟩ ⇒ ⇒ ⟨идентификатор⟩,

⟨фактический параметр⟩ → ⟨выражение⟩ → → ⟨арифметическое выражение⟩ ⇒ ⟨переменная⟩ ⇒ ⇒ ⟨идентификатор⟩,

⟨фактический параметр⟩ → ⟨выражение⟩ → → ⟨арифметическое выражение⟩ ⇒ ⟨указатель функции⟩ ⇒ ⇒ ⟨идентификатор⟩.

Эти синтаксические неоднозначности отражают имеющиеся *семантические* неоднозначности, если рассматриваемый идентификатор является в процедуре формальным параметром, так как в этом случае при отсутствии спецификаций нельзя определить, какой сорт идентификатора будет фактическим аргументом. К тому времени, когда было написано сообщение об Алголе 60, вопрос о синтаксической неоднозначности только еще возник, и авторы этого документа, естественно, обращали мало внимания на ее устранение. Они намеренно ввели в этом примере различия между идентификаторами массива, идентификаторами переключателя и т. д., предусматривая объяснение (со ссылкой на идентификаторы, которые были описаны в соответствующем месте) как часть определения синтаксиса. Ввиду этого в синтаксис Алгола 60 можно включить девятую альтернативу

⟨фактический параметр⟩ → ⟨строка⟩ → → ⟨формальный параметр⟩ → ⟨идентификатор⟩

(так как разд. 4.7.5.1 специально допускает формальные параметры, фактический параметр которых есть строка, используемая как фактический параметр, а этот случай не отражен ни в одной из 8 указанных выше возможностей).

Пропуск девятой альтернативы раскрывает философию сообщения об Алголе 60 в отношении формальных параметров: их следует мысленно заменить фактическими параметрами, *прежде* чем применять синтаксические правила.

Во всяком случае, при проведении анализа предложений (цепочек грамматики) желательно иметь однозначный синтаксис, и, по-видимому, ценой небольших усилий можно переопределить синтаксис Алгола 60 так, чтобы у нас была LR(1)-грамматика для того же языка.

Под языком Алгол 60 мы подразумеваем множество всех цепочек, удовлетворяющих *синтаксису* Алгола 60, но не обязательно удовлетворяющих каким бы то ни было семантическим ограничениям. Например,

begin array $x[100\,000 : 0]$; $y := z/0$ **end.**

Интересно отметить, что, быть может, нельзя определить Алгол 60 с помощью RL(k)-грамматики (RL(k) означает «транслируемость справа налево» и определяется двойственно к LR(k)). Некоторые особенности этого языка делают его наиболее подходящим для чтения слева направо; например, идя слева направо, мы замечаем, что основной символ **comment** радикально меняет анализ букв, стоящих справа от него. Аналогичный язык, для которого существуют LR(k)-грамматики, но не существуют RL(k)-грамматики, рассмотрен в разд. 5 этой статьи. Мы приведем также пример, показывающий, что Алгол 60 *мог бы* быть RL(k).

4. Неразрешимая проблема

Пост [12] доказал неразрешимость знаменитой *проблемы соответствия*, с помощью которой была доказана неразрешимость многих проблем математической лингвистики. Мы докажем здесь неразрешимость проблемы, аналогичной проблеме Поста, и применим полученный результат к изучению LR(k)-грамматик.

Проблема частичного соответствия. Пусть (α_1, β_1), (α_2, β_2), ..., (α_n, β_n) — упорядоченные пары непустых цепочек. Существуют ли для всех $p > 0$ такие упорядоченные наборы из p натуральных чисел $(i_1, i_2, \ldots, i_p)$, что первые p букв цепочки $\alpha_{i_1}\alpha_{i_2} \ldots \alpha_{i_p}$ равны соответственно первым p буквам цепочки $\beta_{i_1}\beta_{i_2} \ldots \beta_{i_p}$?

В обычной проблеме соответствия спрашивается, существует ли число $p > 0$, для которого цепочки $\alpha_{i_1} \ldots \alpha_{i_p}$ и $\beta_{i_1} \ldots \beta_{i_p}$ совпадают.

Хотя неразрешимость проблемы частичного соответствия тесно связана с неразрешимостью обычной проблемы соответствия,

вывести непосредственно одну неразрешимость из другой не удается. Существуют также связи между проблемой частичного соответствия и Tag-проблемой (см. [1]), но выявить эти связи не просто. Однако, применяя методы, аналогичные методам Флойда [6] для обычной проблемы соответствия, и используя детерминированность машин Тьюринга, можно доказать, что проблема частичного соответствия рекурсивно неразрешима.

В нашем доказательстве мы будем пользоваться определением и обозначениями для машин Тьюринга, данными Постом [12]. Мы построим проблему частичного соответствия для любой машины Тьюринга и любой начальной конфигурации. Будем употреблять символы

$$\vdash,\ q_i,\ \bar{q}_i,\ S_j,\ \bar{S}_j,\ h,\ \bar{h},\quad \text{где } 1 \leqslant i \leqslant R \text{ и } 0 \leqslant j \leqslant m.$$

Если начальная конфигурация есть

$$S_{j_1} S_{j_2} \ldots S_{j_{k-1}} q_{i_1} S_{j_k} \ldots S_{j_n},$$

то пара цепочек

$$(\vdash,\ \vdash h S_{j_1} \ldots S_{j_{k-1}} q_{i_1} S_{j_k} \ldots S_{j_n} h) \tag{4.1}$$

входит в нашу проблему частичного соответствия. Добавим также пары

$$(\bar{h},\ h),\ (h,\ \bar{h}),\ (S_j,\ \bar{S}_j),\ (\bar{S}_j,\ S_j)\ (\bar{q}_i,\ q_i),$$
$$\text{где } 1 \leqslant i \leqslant R,\ 0 \leqslant j \leqslant m. \tag{4.2}$$

Наконец, включим пары, определяемые командами машины Тьюринга:

Вид команды	Соответствующие пары, $0 \leqslant t \leqslant m$
$q_i S_j L q_l$	$(h q_i S_j,\ \bar{h} \bar{q}_l \bar{S}_0 \bar{S}_j),\ (S_t q_i S_j,\ \bar{q}_l \bar{S}_t \bar{S}_j)$
$q_i S_j R q_l$	$(q_i S_j h,\ \bar{S}_j \bar{q}_l \bar{S}_0 \bar{h}),\ (q_i S_j S_t,\ \bar{S}_j \bar{q}_l \bar{S}_t)$
$q_i S_j S_k q_l$	$(q_i S_j,\ \bar{q}_l \bar{S}_k)$

(4.3)

Легко видеть, что эти соответствующие пары будут моделировать поведение машины Тьюринга. Поскольку пара (4.1) — это единственная пара, в которой первые буквы совпадают, и поскольку только пары в (4.2) — единственные пары, содержащие некоторые символы q_i в левых цепочках, единственно возможными цепочками, которые могут быть начальными подцепочками в $\alpha_{i_1}\alpha_{i_2}\ldots$ и $\beta_{i_1}\beta_{i_2}\ldots$ одновременно, являются начальные подцепочки последовательности

$$\vdash \alpha_0 \bar{\alpha}_1 \alpha_1 \bar{\alpha}_2 \alpha_2 \bar{\alpha}_3 \alpha_3 \ldots, \tag{4.4}$$

где $\alpha_0, \alpha_1, \alpha_2, \ldots$ представляют последовательные шаги ленты машины Тьюринга (с символами h, помещенными у каждого конца; $\bar{\alpha}$ обозначает цепочку, полученную из α заменой каждой буквы ее двойником). Поэтому для таких пар *проблема частичного соответствия имеет утвердительное решение тогда и только тогда, когда машина Тьюринга вообще не останавливается.* А проблема останова машины Тьюринга, как хорошо известно, рекурсивно неразрешима.

Применим этот результат к LR(k)-грамматикам.

Теорема 4.1 *Проблема определения, существует ли для заданной произвольной грамматики* 𝔊 *такое число* $k \geqslant 0$, *что* 𝔊 *является* LR(k)*-грамматикой, рекурсивно неразрешима.*

Эта теорема контрастирует с результатами разд. 2, где установлена разрешимость аналогичной проблемы при заданном k. Для доказательства сформулированной теоремы сведем проблему частичного соответствия к LR(k)-проблеме для специального класса грамматик.

Пусть $(\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n)$ — пары цепочек, входящие в проблему частичного соответствия, и $X_1, X_2, \ldots, X_n, +$ обозначают $n+1$ букв, отличных от тех, которые встречаются в α и β. Рассмотрим следующую грамматику 𝔊:

$$S \to A,\ S \to B,\ A \to X_i + \alpha_i,\ B \to X_i + \beta_i,$$
$$A \to X_i A \alpha_i,\ B \to X_i B \beta_i,\ 1 \leqslant i \leqslant n. \tag{4.5}$$

Ее выводимыми цепочками будут

$$\{X_{i_m} \ldots X_{i_1} A \alpha_{i_1} \ldots \alpha_{i_m}\} \cup \{X_{i_m} \ldots X_{i_1} B \beta_{i_1} \ldots \beta_{i_m}\} \cup$$
$$\cup \{X_{i_m} \ldots X_{i_1} + \alpha_{i_1} \ldots \alpha_{i_m}\} \cup \{X_{i_m} \ldots X_{i_1} + \beta_{i_1} \ldots \beta_{i_m}\}.$$

Покажем, что 𝔊 является LR(k)-грамматикой для некоторого k тогда и только тогда, когда наша проблема частичного соответствия имеет отрицательное решение. Если решение утвердительное, то для каждого p найдутся выводимые цепочки $X_{i_p} \ldots X_{i_1} + \alpha_{i_1} \ldots \alpha_{i_p}$, $X^{i}_{p} \ldots X_{i_1} + \beta_{i_1} \ldots \beta_{i_p}$, в которых первые p букв, идущих за $+$, совпадают. Основа должна включать символ $+$, но следующие за ней $p-q$ букв, где q — максимальная длина цепочек α_i, β_i, не позволяют определить, какую из продукций $A \to X_{i_1} + \alpha_{i_1}$ или $B \to X_{i_1} + \beta_{i_1}$ надо применить. Следовательно, эта грамматика не является LR($p-q$). С другой стороны, если проблема частичного соответствия решается отрицательно, то существует такое p, для которого по $(i_1, \ldots, i_{\min(p,\,t)})$ и первым p буквам цепочки $\alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_t} \dashv^{p}$ или $\beta_{i_1}\beta_{i_2}\ldots\beta_{i_t} \dashv^{p}$ можно выяснить, какая именно эта цепочка, так что 𝔊 оказывается грамматикой ограниченного контекста.

Мы доказали даже несколько более сильное утверждение, ответив на вопрос, поставленный Флойдом [5]:

Теорема 4.2. *Следующие проблемы рекурсивно неразрешимы*: (i) *является ли данная грамматика грамматикой ограниченного контекста*; (ii) *является ли данная грамматика грамматикой ограниченного правого контекста.*

Эти теоремы можно усилить обычными способами, показав, что если грамматика $\mathfrak{G}$ предполагается однозначной, линейной, имеющей не более двух терминалов, имеющей либо ограниченное число продукций, либо ограниченную длину цепочки в продукциях, то рассматриваемая проблема остается неразрешимой.

5. Связи с детерминированными языками

Гинзбург и Грейбах [7] определили *детерминированный язык* как язык, распознаваемый так называемым *детерминированным магазинным автоматом* (ДМПА). Последний представляет собой устройство с конечным числом (внутренних) состояний $q_0, q_1, \ldots, q_r$, оперирующее с цепочками букв в двух алфавитах T и I в соответствии с правилами следующих двух типов:

$$Aq_i \to \theta q_j, \tag{5.1}$$

$$Aq_i a \to \theta q_j. \tag{5.2}$$

Здесь $A \in I$, $a \in T$, а θ может быть любой цепочкой в алфавите I. Если A обозначает специальный символ $\vdash$, то мы требуем, чтобы цепочка θ была непустой и $\vdash$ был ее начальным символом. Предполагается, что для каждой пары Aq_i, где $A \in I$, $0 \leqslant i \leqslant r$, либо есть единственное правило типа (5.1) и нет ни одного правила типа (5.2), либо нет правил типа (5.1) и для каждого $a \in T$ есть не более одного правила типа (5.2). Некоторые состояния отмечены как *заключительные*, и терминальная цепочка α *допускается* ДМПА тогда и только тогда, когда $\vdash q_0\alpha \Longrightarrow \vdash \omega q_i$ для некоторого заключительного состояния q_i и некоторой цепочки ω. Здесь отношение $\Longrightarrow$ порождается отношением $\to$ так же, как в разд. 1.

Теорема 5.1. *Если* $\mathfrak{G}$ *является* $LR(k)$*-грамматикой и определяет язык* L, *то существует* ДМПА, *распознающий язык* $L\dashv^k$.

Вторая конструкция из разд. 2 фактически тесно связана с ДМПА. Грамматика $\mathfrak{G}$, *пополненная* правилом (2.7), определяет язык $L\dashv^k$. В качестве состояний ДМПА возьмем терминальные k-буквенные цепочки $[Y_1 \ldots Y_k]$ и различные дополнительные состояния. Терминальным алфавитом ДМПА будет

$T \cup \{\dashv\}$, а вспомогательным алфавитом будет $\{\mathscr{P}\} \cup I \cup T \cup \{\vdash\}$. Мы хотим, чтобы ДМПА достигал конфигурации

$$\vdash \mathscr{P}_0 X_1 \mathscr{P}_1 \ldots X_n \mathscr{P}_n [Y_1 \ldots Y_k]\omega \tag{5.3}$$

тогда и только тогда, когда стек в алгоритме анализа из разд. 2 на соответствующем этапе вычисления содержит $\mathscr{P}_0 X_1 \mathscr{P}_1 \ldots X_n \mathscr{P}_n | Y_1 \ldots Y_k \omega$.

Очевидно, что можно составить правила вида (5.2), читающие первые k букв нашей входной цепочки $Y_1 \ldots Y_k\omega$ и приводящие к начальной конфигурации $\vdash \{[0, 0; \dashv^k]\}[Y_1 \ldots Y_k]\omega$. Предположим, что ДМПА попал в конфигурацию (5.3); можно вычислить, как на шагах 1 и 2 алгоритма анализа, множества Z и Z_p. Если $Y_1 \ldots Y_k \in Z$, составим команды вида (5.2):

$$\mathscr{P}_n [Y_1 \ldots Y_k] a \to \mathscr{P}_n Y_1 \mathscr{P}_{n+1} [Y_2 \ldots Y_k a]; \tag{5.4}$$

где $\mathscr{P}_{n+1}$ определяется равенством $X_{n+1} = Y_1$ (или символом a, если $k = 0$) в (2.14). Если $Y_1 \ldots Y_k \in Z_p$, введем новые дополнительные состояния $q^{(0)}, q^{(1)}, \ldots, q^{(2n_p)}$ и положим

$$\begin{aligned} &\mathscr{P}_n [Y_1 \ldots Y_k] \to \mathscr{P}_n q^{(0)}, \\ &\mathscr{P} q^{(2t)} \to q^{(2t+1)}, \quad X_{p\,(n_p - t)} q^{(2t+1)} \to q^{(2t+2)}, \quad 0 \leqslant t < n_p, \\ &\qquad\qquad \text{для всех } \mathscr{P}, \\ &\mathscr{P} q^{(2n_p)} \to \mathscr{P} A_p \mathscr{P}_{n+1} [Y_1 \ldots Y_k] \quad \text{для всех } \mathscr{P}, \end{aligned} \tag{5.5}$$

где $\mathscr{P}_{n+1}$ определяется из $\mathscr{P}$ согласно (2.14) с $\mathscr{P}_n = \mathscr{P}$, $X_{n+1} = A_p$. Сделаем одно исключение из этого правила: если $Y_1 \ldots Y_k = \dashv^k$ и $\mathscr{P} = \{[0, 0; \dashv^k]\}$, то последнюю команду заменим на $\mathscr{P} q^{(2n_p)} \to q_f$, где q_f — единственное заключительное состояние нашего ДМПА.

Правила (5.4) и (5.5) для всех возможных комбинаций множеств $\mathscr{P}_n$ и $[Y_1 \ldots Y_k]$ вместе с некоторым количеством начальных и заключительных состояний дают ДМПА, который в точности следует процедуре алгоритма анализа из разд. 2.

Следствие 5.1. *Если* $\mathfrak{G}$ *является* $LR(k)$*-грамматикой и определяет язык* L, *то существует* ДМПА, *распознающий язык* L.

Гинзбург и Грейбах [7] доказали наряду с другими интересными теоремами, что для детерминированного языка L_0 и регулярного языка R язык $\{\alpha \mid \alpha\beta \in L_0$ для некоторого $\beta \in R\}$ детерминированный. Возьмем $L_0 = L\dashv^k$ и $R = \{\dashv^k\}$.

Докажем обратный результат.

Теорема 5.2. *Если L — детерминированный язык, то существует* LR(1)*-грамматика* Ⓖ, *определяющая* L.

Для доказательства возьмем произвольный ДМПА с командами вида (5.1) и (5.2) и построим соответствующую грамматику. Сначала несколько упростим задачу, а именно потребуем, чтобы все команды ДМПА были одного из трех типов:

$$\begin{aligned} &\text{тип (i):} && Aq_ia \to Aq_j, \\ &\text{тип (ii):} && Aq_i \to q_j, \\ &\text{тип (iii):} && Aq_i \to ABq_j, \end{aligned} \tag{5.6}$$

где A и B — вспомогательные символы, а a — терминальный символ. Это не умеьшает общности, поскольку правила (5.2) можно заменить правилами $Aq_ia \to Aq$ и $Aq \to \theta q_j$ для некоторого нового состояния q, и у нас останутся правила типа (i) и правила вида (5.1). Правило $Aq_i \to \theta q_j$ есть правило типа (ii), если θ — пустая цепочка; в противном случае положим $\theta = A_1A_2 \ldots A_t$, где $t \geqslant 1$. Если $A_1 \neq \mathbf{A}$, то $A \neq \vdash$, так что правило (5.1) можно заменить правилами $Aq_i \to q$, $Bq \to BA_1q'$ для всех вспомогательных символов B и $A_1q' \to \theta q_j$, где q, q' — новые состояния. Таким образом, можно положить $A = A_1$ и, значит, правило (5.1) можно заменить $t-1$ правилами типа (iii), вводя $t-2$ новых состояний и предполагая, что $t > 1$. Наконец, если $t = 1$, то правило $Aq_i \to Aq_j$ можно заменить правилами $Aq_i \to AAq$, $Aq \to q_j$, где q — новое состояние. Тем самым все правила сведены к типам (5.6).

Любая пара Aq_i обладает свойством детерминированности; если больше чем в одном правиле в левую часть входит Aq_i, то все такие правила относятся к типу (i) и для каждой отдельной терминальной буквы a существует не более одного такого правила.

Сделаем ряд дополнительных предположений о *заключительных состояниях*. Если q_f, q'_f — заключительные состояния (возможно, совпадающие), то мы хотим устранить возможность

$$\alpha q_f \Rightarrow \beta q'_f, \tag{5.7}$$

поскольку она привела бы к тому, что входная цепочка «дважды допускалась бы» ДМПА. Чтобы исключить эту возможность, удвоим число состояний в ДМПА, используя для каждого исходного состояния q_i два состояния q_i и $\bar{q}_i$. Команды (5.6) заменим командами

типа (i): $Aq_ia \to Aq_j$, $A\bar{q}_ia \to Aq_j$;

типа (ii): $Aq_i \to q_j$, если состояние q_i *не* заключительное,
$Aq_i \to \bar{q}_j$, если состояние q_i заключительное,
$A\bar{q}_i \to \bar{q}_j$;

типа (iii): $Aq_i \to ABq_j$, если состояние q_i *не* заключительное,
$Aq_i \to AB\bar{q}_j$, если состояние q_i заключительное,
$A\bar{q}_i \to AB\bar{q}_j$.

Легко проверить, что (5.7) не может произойти и что допускается то же самое множество цепочек; по существу мы переходим в состояние $\bar{q}_j$, если рассматриваемая цепочка уже была допущена, и тогда не распознаем эту цепочку снова, а при применении следующего правила типа (i) возвращаемся к состоянию q_j.

Пусть ДМПА после всех описанных изменений имеет состояния $q_0, \ldots, q_r$; теперь можно построить грамматику для языка, распознаваемого этим ДМПА. Начнем с определения языков L_{iAt} для $0 \leqslant i, t \leqslant r$ и всех $A \in I$:

$$L_{iAt} = \{\alpha \mid Aq_i\alpha \equiv\rangle' Aq \to q_t \text{ для некоторого } q\}, \tag{5.8}$$

где ни на одном шаге вывода $\equiv\rangle'$ не затрагивается входящий слева символ A.

Построим продукции для всех правил (5.5) нашего ДМПА:

Правила	Продукции для Ⓖ
тип (i): $Aq_ia \to Aq_j$	$L_{iAt} \to aL_{jAt}$, $0 \leqslant t \leqslant r$
тип (ii): $Aq_i \to q_j$	$L_{iAt} \to \varepsilon$
тип (iii): $Aq_i \to ABq_j$	$L_{iAt} \to L_{jBs}L_{sAt}$, $0 \leqslant s,\ t \leqslant r$.

(5.9)

Индукцией по длине вывода $\equiv\rangle'$ или вывода в грамматике Ⓖ легко установить совпадение множеств цепочек, определяемых в (5.8), и множеств цепочек, выводимых из L_{iAt} с помощью продукций (5.9).

Большое значение имеет также множество

$$L_{iA} = \{\alpha \mid Aq_i\alpha \Longrightarrow' A\omega q_f \text{ для некоторой цепочки } \omega \text{ и некоторого заключительного состояния } q_f\}. \quad (5.10)$$

Запишем следующие продукции:

Правила	Продукции для ⓖ
тип (i): $Aq_ia \to Aq_j$	$L_{iA} \to aL_{jA}$
тип (ii): $Aq_i \to q_j$	нет
тип (iii): $Aq_i \to ABq_j$	$L_{iA} \to A_{jB}$, $L_{iA} \to L_{jBs}L_{sA}$, $0 \leqslant s \leqslant r$
q_i заключительное	$L_{iA} \to \varepsilon$ для всех A

. (5.11)

Опять-таки по индукции легко установить эквивалентность определений (5.10) и (5.11). Язык, выводимый из $L_{0\vdash}$ с помощью продукции грамматики ⓖ, является в точности языком L из доказываемой теоремы.

Теперь удалим из ⓖ все бесполезные продукции, т. е. те, которые никогда не встречаются в выводе терминальной цепочки, начинающемся с $L_{0\vdash}$. Мы утверждаем, что получившаяся грамматика является LR(1). Этот результат можно доказать с использованием построений из разд. 2, где множества положений имеют довольно простой вид, но для наглядности мы приведем более интуитивное изложение[1]), которое покажет связь между действиями ДМПА и процессом анализа.

Рассмотрим любую цепочку $\alpha\dashv$, где α допускается ДМПА, и исследуем шаг за шагом поведение ДМПА в процессе обработки α. При этом мы построим частичное дерево вывода, отражающее всю информацию, которая будет получена на данном этапе анализа. Вершины этого частичного дерева отметим символами $[i, A, *]$, означающими, что в единственном возможном анализе цепочки на этом месте должен стоять нетерминальный символ L_{iAt} при *некотором* $t = 0, 1, \ldots, r$ или L_{iA}. Мы будем обозревать некоторую вершину $[i, A, *]$, т. е. нас интересует эта конкретная вершина, расположенная ниже основы, и в этот момент ДМПА находится в конфигурации $\ldots Aq_i \ldots$.

[1]) Формальное доказательство читатель сможет найти в статье Лемана на стр. 43—46 настоящего сборника. — *Прим. ред.*

Поясним это на примере, рассмотрев следующий «случайно выбранный» ДМПА:

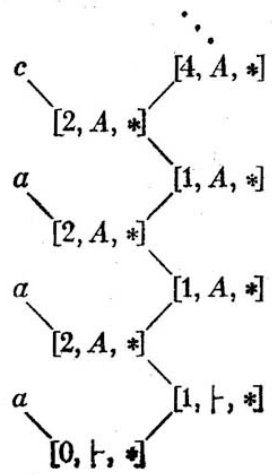
Правила ДМПА	Продукции грамматики ⓖ (исключая некоторые вычеркнутые)
$\vdash q_0a \to \vdash q_1$	$L_{0\vdash} \to aL_{1\vdash}$
$\vdash q_1 \to \vdash Aq_2$	$L_{1\vdash} \to L_{2A}$, $L_{1\vdash} \to L_{2A5}L_{5\vdash}$
$Aq_2a \to Aq_1$	$L_{2At} \to aL_{1At}$ $(t = 2, 5, 6)$, $L_{2A} \to aL_{1A}$
$Aq_1 \to AAq_2$	$L_{1A} \to L_{2A}$, $L_{1A} \to L_{2A2}L_{2A}$,
.	$L_{1A2} \to L_{2A6}L_{6A2}$, $L_{1At} \to L_{2A2}L_{2At}$
$Aq_2b \to Aq_3$	$L_{2A5} \to bL_{3A5}$, $L_{2A} \to bL_{3A}$
$Aq_2c \to Aq_4$	$L_{2A6} \to cL_{4A6}$
$Aq_3 \to q_5$	$L_{3A5} \to \varepsilon$
$Aq_4 \to q_6$	$L_{4A6} \to \varepsilon$
$Aq_6 \to q_2$	$L_{6A2} \to \varepsilon$
$\vdash q_5c \to \vdash q_1$	$L_{5\vdash} \to cL_{1\vdash}$
q_1 заключительное	$L_{1A} \to \varepsilon$, $L_{1\vdash} \to \varepsilon$
q_3 заключительное	$L_{3A} \to \varepsilon$

(5.12)

Рассмотрим действие ДМПА, когда выбрана цепочка $aaacb\dashv$. Имеем

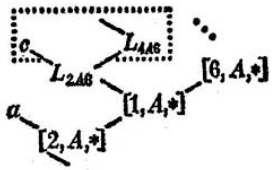
$$\vdash q_0aaacb\dashv \to \vdash q_1aacb\dashv \to \vdash Aq_2aacb\dashv \to \vdash Aq_1acb\dashv \to \to \vdash AAq_2acb\dashv \to \vdash AAq_1cb\dashv \to \vdash AAAq_2cb\dashv \to \to \vdash AAAq_4b\dashv.$$

В соответствии с этими 7 переходами построим частичное дерево, строя на каждом шаге



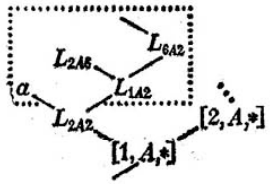
(5.13)

Мы теперь обозреваем вершину $[4, A, *]$, выше которой стоят три точки. В этом месте ДМПА применяет правило $Aq_4 \to q_6$, а мы преобразуем вершину дерева (5.13) в



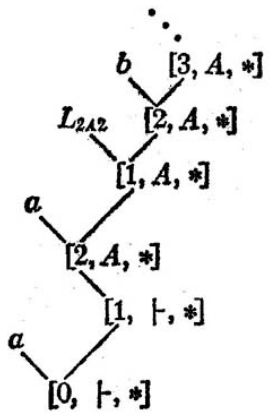
(5.14)

(Таким образом распознаются две основы и затем выбрасываются из дерева.) Далее ДМПА применяет правило $Aq_6 \to q_2$, и (5.14) переходит в



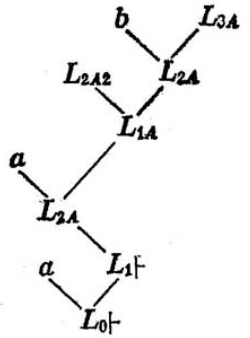
(5.15)

при этом редуцируются еще три основы. В результате применения правила $Aq_2b \to Aq_3$ дерево принимает вид



(5.16)

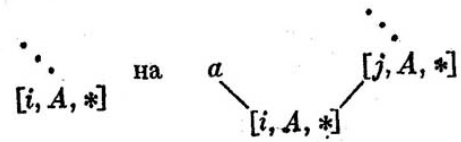
Так как q_3 — заключительное состояние, а следующий символ есть $\dashv$, то анализ завершен; (5.16) переходит в



(5.17)

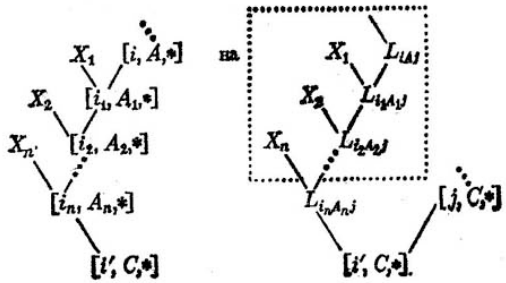
Теперь разберем общий случай. Предположим, что ДМПА находится в конфигурации $\ldots CAq_ia \ldots$, а мы обозреваем вершину $[i, A, *]$ дерева. Если q_i — заключительное состояние, а $a = \dashv$, то, согласно условию (5.7), нам надо выполнить анализ, и мы заменяем каждую вершину $[i, A, *]$ в дереве на L_{iA} до тех пор, пока не достигнем корня (как при переходе от (5.16) к (5.17)). Если состояние q_i не является заключительным или $a \neq \dashv$, то в зависимости от пары Aq_i могут представиться три случая.

Случай (i). ДМПА содержит правило вида $Aq_ia \to Aq_j$. Тогда при замене



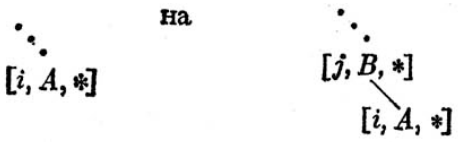
возможен единственный способ анализа, как при переходе от (5.15) к (5.16).

Случай (ii). ДМПА содержит правило вида $Aq_i \to q_j$. Тогда надо заменить наше дерево



как при переходах от (5.13) к (5.14) и от (5.14) к (5.15). Здесь $n \geqslant 0$.

Случай (iii). ДМПА содержит правило вида $Aq_i \to ABq_j$. Тогда при замене



возможен единственный способ анализа, как при построении дерева (5.13).

По определению ДМПА случаи (i), (ii) и (iii) исключают друг друга, и наши рассуждения основаны на том, что дерево представляет все возможные продукции грамматики, которые могли бы работать. Заметим, что в анализе мы *фактически имели дело почти с* LR(0)*-грамматикой*, поскольку для выяснения того, является ли следующая за основой буква маркером $\dashv$, ее нужно было рассматривать только тогда, когда q_i — заключительное состояние.

Из теорем 5.1 и 5.2 следует, что язык может быть порожден LR(k)-грамматикой тогда и только тогда, когда он детерминированный, и тогда и только тогда, когда он порождается некоторой LR(1)-грамматикой. Это утверждение нельзя улучшить, взяв LR(0)-грамматику вместо LR(k) (или LR(1)), так как очевидно, что даже простой язык $\{\varepsilon, a\}$ не может быть порожден LR(0)-грамматикой.

Однако можно доказать, что *язык* $L\dashv$ *всегда может быть порожден некоторой* LR(0)-грамматикой; просто надо взять LR(1)-грамматику из теоремы 5.2 и применить теорему 5.1, чтобы для $L\dashv$ получить другой ДМПА. Этот ДМПА имеет единственное заключительное состояние q_f, из которого нельзя перейти в другие состояния, так что конструкция теоремы 5.2, примененная к этой новой грамматике, дает LR(0)-грамматику. Детерминированный язык, в котором ни одна допустимая цепочка не является собственной начальной подцепочкой какой-нибудь другой допустимой цепочки, может быть аналогичным образом порожден некоторой LR(0)-грамматикой.

Докажем еще одну теорему, из которой будет следовать, что *детерминированность* обладает асимметрией: существуют языки, транслируемые справа налево, но не детерминированные.

Теорема 5.3. *Продукции*

$$S \to Ac,\quad S \to B,\quad A \to aAbb,\quad A \to abb,\quad B \to aBb,\quad B \to ab \qquad (5.18)$$

образуют RL(0)*-грамматику, порождающую недетерминированный язык.*

Доказательство. Терминальными цепочками этого языка будут цепочки вида $a^nb^{2n}c$ и a^nb^n, где $n > 0$. Ясно, что эта грамматика есть RL(0). С другой стороны, предположим, что для этого языка мы смогли построить LR(k)-грамматику. (Трудность составляет, конечно, появление на правом конце цепочки буквы c.) Если рассмотреть выводы бесконечно многих цепочек a^nb^n, то мы найдем вывод, в котором *рекурсивно* встречается нетерминал C, т. е. найдутся нетерминал C и такие цепочки α, φ, μ, ψ, ω, что

$$S \Rightarrow \alpha C\omega \Rightarrow \alpha\varphi C\psi\omega \Rightarrow \alpha\varphi\mu\psi\omega = a^nb^n$$

для некоторого n. Тогда цепочки $\alpha\varphi^t\mu\psi^t\omega$ должны принадлежать этому языку при всех $t \geqslant 0$, а цепочка $\varphi\psi$ непуста в силу однозначности грамматики. Поэтому $\varphi = a^p$, $\psi = b^p$ для некоторого $p > 0$. Отсюда следует, что C не может появиться в выводе ни одной из цепочек вида $a^nb^{2n}c$. Для произвольно большого t рассматриваемый язык содержит цепочки $\alpha\psi^{t+1}\mu\psi^{t+1}\omega = a^{n+pt}b^{n+pt}$, в которых в силу однозначности основа должна иметь по крайней мере $p(t+1)$ букв справа и должна приводить к выводимой цепочке $\alpha\varphi^{t+1}C\psi^{t+1}\omega$ с $p(t+1)$ буквами справа от основы. Кроме того, этот язык содержит цепочки $a^{n+pt}b^{2(n+pt)}c$, у которых *не* может быть той же самой основы, так что грамматика не может быть LR(k). По теореме 5.2 этот язык не является детерминированным при чтении слева направо.

Когда готовилась настоящая статья, автор делал попытку доказать что язык $\{a^nb^n\}\, d \cup (a, b)^* c$ не может быть порожден LR(k)-грамматикой. Вначале это предположение казалось правдоподобным, но на самом деле указанный язык порождается грамматикой

$$\begin{aligned} &S \to A,\ S \to bC,\ S \to Bd,\ S \to BcC,\ S \to c,\\ &A \to Bc,\ A \to BaC,\ A \to aA,\\ &B \to ab,\ B \to aBb,\\ &C \to c,\ C \to aC,\ C \to bC. \end{aligned} \qquad (5.19)$$

Это LR(0)-грамматика.

Действительно, можно заметить, что ДМПА способен распознавать дополнение к множеству допускаемых им цепочек, так что если L — детерминированный язык, не содержащий букву c, то язык $L \cup \{\alpha c \mid \alpha$ — цепочка из терминальных символов,

участвующих в языке L) должен быть детерминированным вопреки ожиданиям. Это ослабляет довод, согласно которому **comment** в Алголе 60 может делать его не RL-языком.

6. Замечания и открытые вопросы

Понятие LR(k)-грамматики вносит определенную ясность в проблему трансляции языков, порождаемых контекстно-свободными грамматиками, и открывает интересные области дальнейших исследований.

Наибольший интерес представляет изучение грамматических преобразований, сохраняющих LR(k)-условие. Многие такие преобразования хорошо известны (например, удаление из грамматики правил с пустой правой частью; исключение левых рекурсий; приведение к нормальной форме, в которой все продукции имеют вид $A \to BC$ или $A \to a$; операция трансдукции, превращающая грамматику в течение ее трансляции в другую грамматику, и много частных случаев последней операции). Какие из этих грамматических модификаций переводят LR(k)-грамматики в LR(k)-грамматики? Аналогичные вопросы встают для грамматик ограниченного контекста и ограниченного правого контекста.

Другую важную область исследования составляет построение алгоритмов, распознающих LR(k)-грамматики или их подклассы, и механическое нахождение программ эффективного анализа. В разд. 3 мы указали три способа упрощения общих схем анализа, порождаемых нашей конструкцией; таких способов, существует, конечно, много. Таблица типа табл. 2 отражает по существу всю информацию, получаемую в процессе анализа; значительную ее часть можно рассматривать как повторяющуюся или избыточную.

Можно сделать также и некоторые выводы, касающиеся теории автоматов. Мы показали, что детерминированный магазинный автомат распознает в точности те языки, которые могут быть порождены LR(k)-грамматиками. Этот результат можно усилить, показав, что в действительности такие языки всегда можно задать с помощью грамматик *ограниченного правого контекста*. Для этого достаточно просто модифицировать конструкцию (5.9), (5.11), заменив продукции

$$L_{iAt} \to \alpha \quad \text{на} \quad L_{iAt} \to M_{iA}\alpha,$$
$$L_{iA} \to \alpha \quad \text{на} \quad L_{iA} \to M_{iA}\alpha$$

и добавив продукции $M_{iA} \to \varepsilon$ для всех i, A. Это сохраняет необходимую информацию в выводимой цепочке, подлежащей анализу.

Однако встает вопрос, какого типа автоматы способны распознавать в точности те языки, которые порождаются грамматиками *ограниченного контекста*. Условие ограниченного контекста симметрично относительно левого и правого, а мы показали, что детерминированность асимметрична; например, зеркальный образ языка (5.18) является детерминированным языком, который не может быть порожден грамматикой ограниченного контекста.

Еще одна интересная область исследования — скорость анализа. Хотя LR(k)-грамматики могут быть проанализированы за время, пропорциональное длине анализируемой цепочки, существуют более общие грамматики, которые могут быть проанализированы с линейной скоростью. Они могут включать, например, возвращение назад ограниченное число раз, просмотры слева направо в комбинации с просмотрами справа налево и т. д.

Неизвестно, существуют ли методы анализа, обеспечивающие линейное время для всех грамматик[1]). Наконец, упомянем еще об одном обобщении LR(k)-грамматик как возможном объекте исследования. Можно рассмотреть вторую основу дерева, т. е. крайнюю слева законченную ветвь терминалов, лежащую справа от основы; аналогично можно ввести понятие r-й основы. Процесс анализа, всегда редуцирующий одну из t первых основ, ведет к грамматикам, которые можно назвать LR(k, t)-грамматиками (в нашем случае $t = 1$). Грамматика

$$S \to ACc, \quad S \to BCd, \quad A \to a, \quad B \to a, \quad C \to Cb, \quad C \to b \tag{6.1}$$

не является LR(k, 1) ни для какого k, так как a в обеих цепочках ab^nc и ab^nd служит основой, но она является LR(0, 2). Для анализа грамматики (6.1) можно использовать следующие правила редукции:

$$ab \to aC, \quad Cb \to C, \quad aCc \to ACc, \quad aCd \to BCd, \quad ACc \to S, \quad BCc \to S.$$

Можно назвать это *трансляцией слева направо*, хотя мы возвращались назад на конечное расстояние.

Список литературы

1. Cocke J., Minsky M., Universality of Tag systems with $P = 2$, *J. Assoc. Comput. Mach.*, 11 (1964), 15—20.

[1]) Относительно времени анализа произвольной контекстно-свободной грамматики, а также времени анализа более широких, чем LR(k), подклассов контекстно-свободных грамматик см. статью Эрли (стр. 47—70 настоящего сборника), а также статью Янгера в сб. «Проблемы математической логики», изд-во «Мир», М., 1970, стр. 344—362. — *Прим. ред.*

2. Earley J., Generating productions from BNF (предварительное сообщение), Carnegie Institute of Technology, 1964.
3. Eickel J., Generation of parsing algorithms for Chomsky type 2 languages, Tech. Hoch. München, Ber. № 6401, 1964.
4. Floyd R. W., Syntactic analysis and operator precedence, *J. Assoc. Comput. Mach.*, **10** (1963), 316—333.
5. Floyd R. W., Bounded context syntactic analysis, *Commun. Assoc. Comput. Mach.*, **7** (1964), 62—66.
6. Floyd R. W., New proofs of old theorems in logic and formal linguistics, Computer Associates, Inc., Wakefield, Massachusetts, 1964.
7. Ginsburg S., Greibach S., Deterministic context-free languages (предварительное сообщение), *Am. Math. Soc. Not.*, **12** (1965), 216, 367.
8. Irons E. T., "Structural connections» in formal languages, *Commun. Assoc. Comput. Mach.*, **7** (1964), 67—71.
9. Lynch W. C., Ambiguities in BNF Languages, Thesis, Univ. of Wisconsin, 1963.
10. Naur P., ed., Revised Algol 60 report, *Commun. Assoc. Comput. Mach.*, **6**, (1963), 1—17.
11. Paul M., A general processor for certain formal languages, Proc. Symp. Symbolic Languages in Data Processing, Rome, 1962, New York, 1962.
12. Post E. L., Recursive unsolvability of a problem of Thue, *J. Symp. Logic*, **12** (1947), 1—11.
13. Гинзбург С., Математическая теория контекстно-свободных языков, изд-во «Мир», М., 1970. (Добавлено при переводе. — *Перев.*)

LR(k)-ГРАММАТИКИ И ДЕТЕРМИНИРОВАННЫЕ ЯЗЫКИ[1])

Дэниел Леман

В этой заметке мы даем четкое доказательство утверждения о том, что для каждого детерминированного языка существует LR(1)-грамматика. Это утверждение впервые было сформулировано Кнутом [3] и воспроизведено затем в монографии Хопкрофта и Улмана [2], но с неудовлетворительным доказательством. Последнее объясняется тем, что грамматика, о которой утверждается, что она есть LR(k)-грамматика, получается из моделирования по Гинзбургу [1] магазинного автомата (МПА), т. е. этот МПА моделирует левосторонние выводы построенной грамматики. На самом же деле LR(k)-грамматики определяются свойством своих правосторонних выводов. Переход от левосторонних выводов к правосторонним не легкий, и неясно, как можно было бы исправить доказательство в [2].

Мы будем пользоваться эквивалентным определением LR(k)-грамматик, указанным Льюисом и Стирнзом [4] и не связанным с правосторонними выводами. Мы докажем эквивалентность этих двух определений, так как, насколько нам известно, такое доказательство еще не было опубликовано.

Наши обозначения совпадают с обозначениями в монографии Хопкрофта и Улмана [2]:

$a, b, c \ldots$	терминалы;
$A, B, C, \ldots$	нетерминалы;
$w, x, y \ldots$	цепочки (слова) из терминалов;
$\alpha, \beta \ldots$	цепочки из терминалов и нетерминалов;
$\rightarrow$	продукция (правило);
$\Rightarrow$	непосредственная выводимость;
$\Rightarrow^*$	транзитивное замыкание для $\Rightarrow$;
$\Rightarrow^*_{rt}$	правосторонняя выводимость;
$\Rightarrow^*_{lf}$	левосторонняя выводимость;
$S \dashv^i$	символ S с приписанными к нему справа i маркерами;
$k: w$	цепочка из k первых букв цепочки w.

[1]) Lehmann D., LR(k)-grammars and deterministic languages, *Israel J. Math.*, **10** (1971), 526—530.

Теорема 1. *Для произвольно выбранной приведенной*[1]) *контекстно-свободной грамматики G два следующих свойства эквивалентны и каждое из них характеризует* LR(k)*-грамматику:*

(а) *Определение Кнута. Грамматика G такова, что если* $S \dashv^k \Rightarrow^*_{rt} \alpha A w_3 \Rightarrow \alpha\beta w_3 (A \to \beta)$ *и* $S \dashv^k \Rightarrow^*_{rt} \alpha\beta w_3'$ *с* k: $w_3' = k$: w_3, *то правосторонний вывод* $\alpha\beta w_3'$ *из* $S \dashv^k$ *обязательно имеет вид* $S \dashv^k \Rightarrow^*_{rt} \alpha A w_3' \Rightarrow \alpha\beta w_3'$.

(б) *Определение Льюиса и Стирнза. Грамматика G однозначна, и если* $S \dashv^k \Rightarrow^* w_1 A w_3$, $A \Rightarrow^* w_2$, $S \dashv^k \Rightarrow^* w_1 w_2 w_3'$ *с* k: $w_3 = k$: w_3', *то* $S \dashv^k \Rightarrow^* w_1 A w_3'$.

Доказательство. (а)$\Rightarrow$(б). В предложении (а) утверждается, что восходящие анализы двух выводимых цепочек (полученные путем правого вывода) совпадают до тех пор, пока в обеих цепочках сразу же справа от части цепочки, редуцируемой в данный момент («основы» по терминологии Кнута), стоят одинаковые подцепочки из k символов. Это обеспечивает однозначность, а если

$$S \dashv^k \Rightarrow^* w_1 A w_3 \Rightarrow^* w_1 w_2 w_3;$$
$$S \dashv^k \Rightarrow^* w_1 w_2 w_3'; \quad k\text{: } w_3 = k\text{: } w_3'$$

и направленные слева направо анализы цепочек $w_1 w_2 w_3$ и $w_1 w_2 w_3'$ совпадают по крайней мере до редукции цепочки w_2 к A, то

$$S \dashv^k \Rightarrow^*_{rt} \alpha A w_3' \Rightarrow^*_{rt} \alpha w_2 w_3' \Rightarrow^*_{rt} w_1 w_2 w_3', \quad S \dashv^k \Rightarrow^* w_1 A w_3'.$$

(б)$\Rightarrow$(а).

$$S \dashv^k \Rightarrow^*_{rt} \alpha A w_3 \Rightarrow \alpha\beta w_3, \quad S \dashv^k \Rightarrow^*_{rt} \alpha\beta w_3' \text{ с } k\text{: } w_3 = k\text{: } w_3'.$$

Поскольку грамматика G приведена, существуют такая цепочка w_1, что $\alpha \Rightarrow^* w_1$, и такая цепочка w_2, что $\beta \Rightarrow^* w_2$. Поэтому

$$S \dashv^k \Rightarrow w_1 A w_3, \quad A \Rightarrow^* w_2, \quad S \dashv^k \Rightarrow w_1 w_2 w_3',$$

и из (б) следует, что $S \dashv^k \Rightarrow^* w_1 A w_3'$.

Итак, у нас есть два способа вывода $w_1 w_2 w_3'$:

$$S \dashv^k \Rightarrow^*_{rt} \alpha\beta w_3' \Rightarrow^*_{rt} \alpha w_2 w_3' \Rightarrow^*_{rt} w_1 w_2 w_3'$$

и

$$S \dashv^k \Rightarrow^* w_1 A w_3' \Rightarrow w_1 \beta w_3' \Rightarrow^* w_1 w_2 w_3'.$$

[1]) То есть такой, в которой устранены тупиковые нетерминалы. Это не уменьшает общности. — *Прим. ред.*

Так как грамматика G однозначна, то эти выводы имеют одинаковые деревья вывода и β выводится из A (в этом дереве):

$$S \dashv^k \Rightarrow^*_{rt} \alpha A w_3' \Rightarrow_{rt} \alpha\beta w_3' \Rightarrow^*_{rt} w_1 w_2 w_3',$$

что и требовалось доказать.

Теорема 2. *Каждый язык, распознаваемый детерминированным магазинным автоматом* (ДМПА), *допускающим цепочку при пустом магазине, может быть задан* LR(0)*-грамматикой.*

Доказательство. Воспользуемся моделированием по Гинзбургу [1] МП-автомата контекстно-свободной грамматикой и докажем, что в детерминированном случае эта грамматика обладает свойством (б), сформулированным в теореме 1.

Предварительное замечание. В моделировании применяются нетерминалы, являющиеся тройками $[p, A, q]$, где p и q — состояния МПА, а A — буква из магазинного алфавита, и язык $L_{[p, A, q]}$, порожденный тройкой $[p, A, q]$, является множеством цепочек, которые ведут МПА из состояния p с единственным символом A в магазине в состояние q с пустым магазином. Наше замечание состоит в том, что в детерминированном случае язык $L_{[p, A, q]}$ имеет свойство префикса: ни один собственный префикс (начало) слова из $L_{[p, A, q]}$ не принадлежит $L_{[p, A, q]}$ и, кроме того, языки $L_{[p, A, q]}$ и $L_{[p, A, r]}$ при $q \neq r$ префиксно-несравнимы, т. е. ни один префикс (в слабом смысле) языка $L_{[p, A, q]}$ не принадлежит $L_{[p, A, r]}$ и обратно. Перейдем к доказательству.

1. Заметим, что грамматика однозначна. Рассмотрим два левосторонних вывода слова x и первое место, где они различаются:

$$S \Rightarrow^*_{lf} w[p, A, q]\alpha,$$
$$[p, A, q] \to a[r, B_1, s_1] \ldots [s_{n-1}, B_n, s_n],$$
$$[p, A, q] \to b[t, C_1, u_1] \ldots [u_{m-1}, C_m, u_m].$$

(i) Если $a = \varepsilon$, то состояние p и магазинный символ A вынуждают ε-движение и тогда $b = \varepsilon$, а оба вывода соответствуют одному и тому же движению ДМПА, так что $r = t$, $n = m$, $B_i = C_i$, $1 \leqslant i \leqslant n$. Если бы было $u_i \neq s_i$ при некотором i, то для наименьшего из целых чисел i, удовлетворяющих неравенству $u_i \neq s_i$, из символов $[s_{i-1}, B_i, s_i]$ и $[u_{i-1}, C_i, u_i] = [s_{i-1}, B_i, u_i]$ выводились бы два слова, одно из которых было бы собственным началом другого[1]), что невозможно. Отсюда следует, что $s_i = u_i$ для $1 \leqslant i \leqslant n$.

[1]) Свойство префиксности заставляет нетерминалы $[s_{j-1}, B_j, s_j]$ порождать одинаковые цепочки в обоих выводах. — *Прим. ред.*

(ii) Если $a \neq \varepsilon$, то $b = a$ — очередная буква в x и, как в (i), продукции соответствуют одному и тому же движению ДМПА.

2. Если $S \Rightarrow^* w_1[p, A, q]w_3$, $[p, A, q] \Rightarrow^* w_2$ и $S \Rightarrow^* w_1 w_2 w_3'$, то существует левосторонний вывод:

$$S \Rightarrow^*_{\text{lf}} w_1[p, A, q][q, B_1, s_1] \ldots [s_{n-1}, B_n, s_n].$$

Это значит, что после прочтения w_1 и нескольких ε-движений ДМПА пришел в состояние p с магазином $AB_0 \ldots B_n$. Тогда в выводе цепочки $w_1 w_2 w_3'$ найдется стадия

$$S \Rightarrow^*_{\text{lf}} w_1[p, A, t][t, B_1, t_1] \ldots [t_{n-1}, B_n, t_n].$$

$[p, A, t] \Rightarrow^* w_2'$, так что w_2' есть префикс цепочки w_2 или w_2 есть префикс цепочки w_2'. Это показывает, что $w_2 = w_2'$ и $t = q$ (согласно нашему предварительному замечанию). Итак,

$$S \Rightarrow^*_{\text{lf}} w_1[p, A, q]\beta \Rightarrow w_1 w_2 w_3',$$

что и требовалось доказать.

Из теоремы 1 вытекают два следствия.

Следствие 1. *Если L — детерминированный язык, то $L \dashv$ определяется некоторой* LR(0)*-грамматикой (так как $L \dashv$ распознается* ДМПА, *допускающим цепочку при пустом магазине*).

Следствие 2. *Если L — детерминированный язык, то его можно задать некоторой* LR(1)*-грамматикой.*

Список литературы

1. Гинзбург С., Математическая теория контекстно-свободных языков, изд-во «Мир», М., 1970.
2. Hopcroft J. E., Ullman S. D., Formal languages and their relation to automata, Addison-Wesley, 1969.
3. Knuth D. E., On the translation of languages from left to right, *Information and Control*, 8 (1965), 607—639. (Русский перевод см. в настоящем сборнике, стр. 9—42).
4. Lewis R. M., II, Stearns R. E., Syntax-directed transductions, *J. Assoc. Comput. Mach.*, 15 (1968), 465—485.

СОДЕРЖАНИЕ