Общая характеристика работы

Актуальность темы работы

В последнее время большое значение приобрела задача построения синтаксических анализаторов для, так называемых, открытых контекстно-свободных языков, т.е. языков, задаваемых контекстно-свободной грамматикой, которая может быть расширена путем добавления новых сущностей, таких как нетерминальные и терминальные символы, а также правила грамматики. И расширение это может быть произведено произвольно между исполнениями алгоритма синтаксического анализа. Подобная ситуация часто встречается в языках интерфейса к системам представления знаний [2-4]. В таких системах множество понятий может расширяться, и вместе с каждым новым понятием удобно также вводить новые синтаксические элементы языка интерфейса данной системы, связанные с добавляемым понятием. Таким образом, расширение языка интерфейса производится вместе с расширением понятийной базы и позволяет пользователю общаться с системой на языке, специализированном в соответствии с областью знаний, в которой данный пользователь производит моделирование. В этом случае, алгоритм синтаксического анализа не только помогает системе «понять» команды пользователя, но и отражает различные изменения понятийной базы. Поэтому, имеет смысл говорить об открытых интерфейсных языках, или об интерфейсных языках открытого типа. Такие языки имеют следующие особенности:

1. В силу того, что язык является интерфейсным, входная терминальная строка для алгоритма синтаксического анализа имеет сравнительно небольшую длину.
2. В силу открытости языка и алгоритмической неразрешимости задачи распознавания однозначности КС-грамматики никакие ограничения на входную КС-грамматику наложить невозможно.
3. В силу возможности расширения интерфейсного языка новыми сущностями, входная для алгоритма синтаксического анализа КС-грамматика часто имеет очень большое число сущностей по сравнению с длиной входной строки.

Синтаксический анализатор можно представить как вычислимую функцию *Parse* двух аргументов:

- КС-грамматики $G = \{N, T, P, S\}$, где N - нетерминальные символы языка, T - терминалы, P - правила языка и S - стартовый нетерминальный символ грамматики.
- Входной строки $\omega = a_1 ... a_n$ длины $|\omega| = n$, где $a_i \in T$ $1 \le i \le n$.

Функция *Parse* производит построение и выдает в качестве результата множество деревьев вывода $Tr_{G\omega}$ (или единственное дерево, если грамматика G однозначная) входной строки ω в грамматике G, если она принадлежит языку $L(G)$, заданному грамматикой G, или *false* - в противном случае.

В традиционной задаче построения синтаксического анализатора, см. например [1] (в дальнейшем мы будем следовать терминологии, введенной в данной книге), при вызовах функции *Parse* меняется только один параметр – входная строка ω. Функция *Parse* в этом случае имеет вид $Parse_G(\omega)$. Поэтому анализ производительности естественно было представлять как оценку функции $Ev_G(n)$, где n - длина входной цепочки ω, т.е. как зависимость производительности алгоритма от длины входной строки. Так как для данной контекстно-свободной грамматики $G = \{N, T, P, S\}$ может существовать много различных строк длины n, то мы разбиваем множество терминальных строк T^+ на классы ω^n равных

по длине строк. Каждый такой класс ω^n терминальных строк длины $|\omega| = n$ и представляется числом n. Функция $Ev_G(n)$, определенная на множестве натуральных чисел N^+, берет n в качестве параметра и выдает максимальное время анализа, выраженное в элементарных операциях, которые алгоритм анализа производит над сущностями грамматики G.

Главной целью данной работы является выяснение особенностей синтаксического анализа открытых интерфейсных контекстно-свободных языков. Эти особенности, очевидно, определяются свойствами языков, для которых производится синтаксический анализ. Эти свойства, в свою очередь, должны проявляться в особенностях грамматик, задающих данные языки. Выше были перечислены основные особые свойства таких грамматик. Но этого недостаточно, необходимо строго определить параметры грамматики, определяющей открытый интерфейсный язык, имеющие влияние на производительность алгоритма синтаксического анализа. Такие параметры будут определены немного позже: это объем грамматики, ее ветвистость и протяженность. Сейчас же нас интересует вопрос, каким образом можно выразить степень влияния этих параметров на алгоритм синтаксического анализа? Наиболее естественным представляется выразить степень влияния в параметрах вычислительной сложности алгоритма синтаксического анализа, что и сделано в настоящей работе. Для выяснения степени влияния особенностей открытого интерфейсного языка на алгоритм синтаксического анализа проведено исследование вычислительной сложности алгоритма определенной выше вычислимой функции $Parse$. Для того, чтобы не отвлекаться на несущественные детали, мы зафиксируем второй параметр функции $Parse$ - входную строку $\omega = a_1 ... a_n$. Поэтому функция $Parse(G, \omega)$ от двух параметров примет вид $Parse_\omega(G)$ от единственного параметра – входной контекстно-свободной грамматики G.

Таким образом, оказывается, что для выяснения особенностей синтаксического анализа открытых интерфейсных контекстно-свободных языков, необходимо произвести оценку вычислительной сложности каждого алгоритма синтаксического анализа, который может быть использован для анализа открытых интерфейсных языков. Для этого сначала попытаемся выяснить, какие из известных в настоящее время алгоритмов синтаксического анализа подходят для анализа открытых интерфейсных языков. Затем из них выберем наиболее подходящий и попытаемся адаптировать его так, чтобы новый полученный алгоритм являлся, по возможности, наиболее оптимальным для синтаксического анализа открытых интерфейсных языков. На основании проведенной в данной работе оценки было выбрано два алгоритма: алгоритм Эрли [5-6] и алгоритм Кока-Янгера-Касами [8, 13]. Для использования алгоритма Кока-Янгера-Касами необходимо привлечь еще несколько алгоритмов, поэтому в данной работе приведено пять алгоритмов семейства алгоритма Кока-Янгера-Касами и проведена оценка их вычислительной сложности в терминах описанных выше параметров входной грамматики.

Особая ситуация возникла с алгоритмом Эрли. Данный алгоритм не строит деревьев вывода входной строки, но дает ответ лишь на вопрос выводится или нет данная входная строка в данной грамматике. Для наших же целей необходимо произвести построение всех деревьев вывода данной строки. Для этого в данной работе строится модификация алгоритма Эрли таким образом, чтобы он производил построение всех деревьев вывода входной строки во время своего исполнения. Заметим сразу, что эта задача достаточно трудна, и особенно, для неоднозначных грамматик. Достаточно отметить, что в разные годы ее пытались решить разные исследователи. Например, в начале 80-х годов Масари Томита пытался решить проблему построения всех деревьев вывода входной строки для неоднозначных грамматик, задающих естественные языки [12]. Это ему не удалось, поэтому Томита произвел модификацию известного алгоритма $LR(k)$-анализа [9]. Данная модификация теперь известна как алгоритм Томиты или алгоритм GLR [11-12]. Известные в данное время реализации алгоритма Эрли [5-7] не производят построения дерева вывода входной строки вообще или производят такое построение только для однозначных грамматик. В данной

работе представлена модификация алгоритма Эрли, позволяющая строить все деревья вывода входной строки. Данная модификация названа адаптированным для построения деревьев вывода алгоритмом Эрли. В работе также произведена оценка вычислительной сложности функции $Parse_{\omega}(G)$, реализованной посредством адаптированного алгоритма Эрли, в терминах упоминавшихся выше параметров входной грамматики.

Цели диссертационной работы

Основное содержание работы состоит в исследовании особенностей синтаксического анализа открытых интерфейсных контекстно-свободных языков. Цели этого исследования заключаются в следующем:

- На основе проведенного исследования особенностей синтаксического анализа открытых интерфейсных контекстно-свободных языков выделить метод оценки эффективности работы алгоритмов синтаксического анализа для таких языков. В качестве главного критерия такой оценки была выбрана вычислительная сложность алгоритма как функция от некоторых, определяемых в диссертационной работе, параметров входной контекстно-свободной грамматики.
- Используя найденный метод оценки эффективности, провести сравнительный анализ существующих алгоритмов, подходящих для синтаксического анализа открытых интерфейсных контекстно-свободных языков.
- На основании проведенной оценки алгоритмов синтаксического анализа, выделить наиболее подходящий для синтаксического анализа открытых интерфейсных контекстно-свободных языков алгоритм синтаксического анализа.

На основании вышеуказанных целей диссертационной работы были поставлены следующие задачи:

- Определить те параметры контекстно-свободной грамматики, которые влияют на вычислительную сложность алгоритмов синтаксического анализа. Таким образом, были выделены следующие параметры контекстно-свободной грамматики, влияющие на вычислительную сложность алгоритмов синтаксического анализа: объем грамматики, ее ветвистость и протяженность.
- Выделить алгоритмы, подходящие для синтаксического анализа открытых интерфейсных контекстно-свободных языков, и провести оценку их вычислительной сложности как функции от объема, ветвистости и протяженности входной контекстно-свободной грамматики. Таким образом, в качестве основных кандидатов, были выделены алгоритм Кока-Янгера-Касами и алгоритм Эрли. Также была проведена оценка их вычислительной сложности как функции от выделенных ранее параметров входной грамматики.
- Если при выборе наиболее подходящего для синтаксического анализа открытых интерфейсных контекстно-свободных языков алгоритма не будет найден подходящий, то попытаться создать новый алгоритм либо путем модификации какого-либо из известных алгоритмов синтаксического анализа, либо новый, приспособленный специально для анализа этого типа языков. С этой целью в работе был описан адаптированный алгоритм Эрли, а также было проведено доказательство корректности работы данного алгоритма.
- После выбора наиболее подходящего для синтаксического анализа открытых интерфейсных контекстно-свободных языков алгоритма, необходимо реализовать его в виде системы динамической компиляции, которая позволит производить синтаксический анализа языков, задаваемых пополняющимися КС-грамматиками.

Методы исследований

Одним из основных методов исследования был анализ существующих алгоритмов синтаксического анализа на предмет широты их применимости, возможности построения выводов входных строк во входной для алгоритма контекстно-свободной грамматике и

производительности алгоритма как функции от выделенных на этапе исследования свойств КС-грамматики.

Другим методом исследования был метод оценки вычислительной сложности алгоритмов синтаксического анализа, где в качестве параметром для оценки выступали специфические свойства входных контекстно-свободных грамматик. Эти свойства также были определены в данной работе и есть объем, ветвистость и протяженность КС-грамматики.

И наконец, в работе используются методы описания формальных грамматик и описания алгоритмов синтаксического анализа для доказательства корректности работы алгоритмов.

Научная новизна

Все результаты диссертации являются новыми. Новым также является подход к задаче оценки вычислительной сложности алгоритмов синтаксического анализа. В качестве параметров оценки вычислительной сложности алгоритмов в данной работе выступает не длина входной строки, как это обычно производится при представлении новых алгоритмов синтаксического анализа, а параметры, выражающие характеристики входной грамматики. Определение таких характеристик также являлось целью данной работы и, в качестве результата, были выделены три характеристики: объем грамматики, ее ветвистость и протяженность.

В результате исследования был выделен наиболее подходящий для синтаксического анализа открытых интерфейсных языков алгоритм синтаксического анализа. Это алгоритм Эрли для распознавания выводимости входной терминальной строки во входной грамматике. Но, так как данный алгоритм не строит деревья вывода входной строки, а только отвечает, выводится или нет данная строка в данной грамматике, то была произведена модификация данного алгоритма и, таким образом, был получен новый алгоритм, названный адаптированным для построения деревьев вывода алгоритмом Эрли. Данный алгоритм применим к анализу неоднозначных контекстно-свободных языков. В работе впервые доказана корректность работы алгоритма, построенного автором.

Новым результатом также является полученная в работе оценка вычислительной сложности алгоритмов синтаксического анализа как зависимости от параметров входной контекстно-свободной грамматики.

Практическая и теоретическая ценность

Диссертационная работа имеет теоретический характер. В процессе исследования проведен обзор существующих в настоящий момент алгоритмов синтаксического анализа и предложен новый алгоритм, наиболее подходящий для синтаксического анализа открытых контекстно-свободных языков. На основе полученных в данной работе теоретических результатов была реализована система динамической компиляции. До настоящего времени не существовало подходящего алгоритма, позволяющего производить синтаксический анализ открытых контекстно-свободных языков. Поэтому, результаты, полученные в данной работе предоставляют инструмент для построения анализаторов для открытых КС-языков.

В данной работе также был предложен метод оценки вычислительной сложности алгоритмов синтаксического анализа и алгоритмов, производящих различные преобразования контекстно-свободных грамматик. На основании этого метода была произведена оценка нескольких, наиболее подходящих для анализа открытых интерфейсных языков, алгоритмов синтаксического анализа. Данный метод дает возможность оценки применимости тех или иных алгоритмов синтаксического анализа для анализа контекстно-свободных языков, имеющих большое число сущностей.

В работе была реализована система динамической компиляции на основе адаптированного для построения деревьев вывода алгоритма Эрли. Система представляет собой синтаксический анализатор с разделением процесса синтаксического анализа на два процесса: лексического и синтаксического анализа, на основе динамически пополняемых множества лексических типов и множества сущностей входной грамматики. Данная система

может применятся как генератор синтаксических анализаторов для динамически пополняемых языков, а также как синтаксический анализатор для языков интерфейса программных систем.

Апробация работы

Результаты работы были доложены:

на семинаре отдела систем математического обеспечения Вычислительного Центра РАН под руководством д. ф.-м. н. Серебрякова В. А.

на семинаре отделения научных исследований по проблемам информатики Всероссийского института научной и технической информации под руководством д. ф. н. Гиляревского Р. С.

Публикации

Основные результаты данной работы опубликованы в работах [1-4] списка опубликованных работ.

Объем работы

Диссертация содержит 140 страниц и состоит из введения, трех глав и списка литературы, содержащего 56 названий.

Содержание работы

Краткое содержание работы

Диссертация включает в себя введение и три главы. В первой главе представлены два алгоритма синтаксического анализа: адаптированный для построения деревьев вывода алгоритм Эрли и алгоритм обхода сверху вниз и слева направо деревьев вывода, построенных в результате работы адаптированного для построения деревьев вывода алгоритма Эрли. В первой главе также представлено доказательство корректности этих алгоритмов. Во второй главе производится оценка вычислительной сложности двух алгоритмов, введенных в первой главе, и семейства из пяти алгоритмов, связанных с алгоритмом Кока-Янгера-Касами. В качестве параметров для оценки вычислительной сложности используются такие параметры входной грамматики, как ее объем, ветвистость и протяженность. На основании проведенной оценки выбирается алгоритм для реализации разборщика. Таким алгоритмов является адаптированный для построения деревьев вывода алгоритм Эрли. Третья глава посвящена методам реализации динамической системы компиляции, основанной на адаптированном для построения деревьев вывода алгоритме Эрли.

Целью данной работы является исследование особенностей синтаксического анализа открытых интерфейсных контекстно-свободных языков. Язык, заданный контекстно-свободной грамматикой, называется открытым, если он может быть пополнен новыми сущностями: правилами грамматики, нетерминальными и терминальными символами, между двумя исполнениями алгоритма синтаксического анализа. Язык называется интерфейсным, если он используется для осуществления взаимодействия между пользователем некоторой системы и этой системой. Таким образом, алгоритм синтаксического анализа используется системой как один из инструментов, позволяющих «понять» предложения пользователя.

Были выделены три основных особенности синтаксического анализа открытых интерфейсных контекстно-свободных языков.

1. В силу того, что язык является интерфейсным, входная терминальная строка для алгоритма синтаксического анализа имеет сравнительно небольшую длину.

2. В силу открытости языка и алгоритмической неразрешимости задачи распознавания является ли данная КС-грамматика однозначной или нет, никакие ограничения на входную КС-грамматику наложить невозможно.
3. В силу возможности расширения интерфейсного языка новыми сущностями, входная для алгоритма синтаксического анализа КС-грамматика часто имеет очень большое число сущностей.

Таким образом, из второй особенности следует, что алгоритмы синтаксического анализа, применимые к открытым интерфейсным контекстно-свободным языкам, должны иметь возможность проводить анализ языков, заданных любой контекстно-свободной грамматикой без каких-либо ограничений. Так как языки используются в качестве интерфейса, то входные строки для алгоритма синтаксического анализа обычно являются достаточно короткими. Данное обстоятельство позволяет включить в рассмотрение алгоритмы, вычислительная сложность которых, оцениваемая функцией от длины входной строки, довольно высока, что не позволяет их использовать в традиционных синтаксических анализаторах.

Третья отличие открытых интерфейсных языков от обычных контекстно-свободных языков заключается в том, что алгоритм синтаксического анализа должен будет работать с массивными грамматиками, которые задаются очень большим множеством сущностей. Данное обстоятельство позволяет поставить проблему оценки вычислительной сложности алгоритмов синтаксического анализа как функции не от длины входной строки, а как функции от «объема» входной грамматики. Кроме того, хотелось бы явно определить свойства входной грамматики, которые имеют влияние на производительность алгоритмов синтаксического анализа. Основным методом, который применялся для такого определения, было рассмотрение работы алгоритмов синтаксического анализа и выделение параметров входной грамматики, которые эти алгоритмы используют. В процессе такого исследования были выделены три таких параметра: ветвистость, протяженность и объем входной грамматики. Ветвистостью контекстно-свободной грамматики называется максимальное число из множества чисел, каждое из которых обозначает количество правил с одним и тем же нетерминалом в левой части. Протяженностью грамматики называют максимальную из длин правых частей правил грамматики. Объем грамматики определяется как сумма числа нетерминалов и правил данной грамматики. Количество терминальных символов в большинстве алгоритмов синтаксического анализа, применимых к открытым интерфейсным языкам, определяющей роли не играет.

Определение 1. Пусть дана КС-грамматика $G=\{N,T,P,S\}$, где N - нетерминальные символы грамматики, T - терминалы грамматики, P - правила грамматики и S - начальный символ грамматики. Положим $|G|=|N|+|P|$, где $|N|$ - количество нетерминалов и $|P|$ - количество правил грамматики. Величину $|G|$ будем называть объемом грамматики G.

Определение 2. Пусть $G=\{N,T,P,S\}$ КС-грамматика и F_A - множество правил грамматики G с нетерминальным символом A в левой части. Обозначим через F_P максимум из $|F_A|$ для всех $A\in N$, где $|F_A|$ - мощность множества F_A. Назовем величину F_P ветвистостью грамматики G.

Нетрудно видеть, что ветвистость F_P, число нетерминальных символов $|N|$ и число правил грамматики $|P|$ связаны соотношениями: $|N|\le|P|\le|N|\times F_P$.

Определение 3. Пусть $G=\{N,T,P,S\}$ КС-грамматика и $M_P=Max\{|\alpha|: A\to\alpha\in P\}$ - максимальная из длин правых частей правил грамматики. Назовем величину M_P протяженностью грамматики G.

Во второй главе производится оценка вычислительной сложности адаптированного для построения деревьев вывода алгоритма Эрли и находится, что сложность данного алгоритма есть $O\left(F_P \times |P|^2 \times M_P^2\right)$ для однозначных грамматик. Отсюда, и из того, что $|P| \le N \times F_P$, следует, что вычислительная сложность алгоритма адаптированного для построения деревьев вывода алгоритма Эрли есть также $O\left(F_P^3 \times |N|^2 \times M_P^2\right)$. Из приведенной оценки видно, что ветвистость грамматики менее влияет на производительность алгоритма синтаксического анализа, чем длина правил грамматики и количество ее правил. С другой стороны, естественно считать, что максимальная длина правила (протяженность грамматики) M_P не превышает максимальную длину входного слова $|\omega| = n$. Оценка вычислительной сложности для неоднозначной грамматики есть $O\left(F_P \times |P|^2 \times M_P^{2+|\omega|}\right)$ или $O\left(F_P^3 \times |N|^2 \times M_P^{2+|\omega|}\right)$. Поэтому, при анализе неоднозначных языков для лучшей производительности алгоритма синтаксического анализа предпочтительнее преобразовать исходную грамматику таким образом, чтобы длины правил были как можно меньше.

Так как исходная КС-грамматика может меняться от одного исполнения алгоритма синтаксического анализа к другому, необходимо минимизировать расходы на дополнительные действия, которые часто производятся в синтаксических анализаторах еще до исполнения самого алгоритма синтаксического анализа. При анализе языков, задаваемых фиксированной КС-грамматикой, часто бывает удобно, еще до исполнения алгоритма синтаксического анализа, выделить из исходной грамматики некоторую информацию, с тем, чтобы не генерировать ее заново каждый раз при очередном исполнении алгоритма синтаксического анализа. Такую модель синтаксического анализа можно назвать статической моделью (СМ). При анализе нерасширяемых языков время, затрачиваемое на выделение подобной информации из входной КС-грамматики, не учитывается, ввиду того, что такая информация выделяется только один раз. В нашем случае, ввиду того, что входная КС-грамматика может меняться от одного запуска синтаксического анализатора к другому, это время учитывать необходимо или, хотя бы, учитывать время, затрачиваемое на модификацию дополнительной информации при добавлении новых сущностей в исходную КС-грамматику. Если это время велико по отношению со временем работы синтаксического анализатора, то такой метод для наших целей неприемлем. Будем называть модель синтаксического анализа, в которой никакой информации о языке до исполнения алгоритма синтаксического анализа не выделяется, динамической моделью (ДМ). Очевидно, что для наших целей построения синтаксического анализатора для открытых контекстно-свободных языков, необходимо использовать динамическую модель компиляции. В диссертационной работе был проведен анализ известных алгоритмов синтаксического анализа и, на основании этого анализа, были отобраны два алгоритма синтаксического анализа: алгоритм Кока-Янгера-Касами и алгоритм Эрли.

Итак, из множества алгоритмов синтаксического анализа, применимых к открытым интерфейсным языкам, необходимо было выделить наиболее подходящий. Для этого необходимо было провести оценку вычислительной сложности алгоритмов как функции от определенных выше параметров входной грамматики и, на основании этой оценки выбрать наилучший алгоритм. Такая оценка была проведена для двух алгоритмов: алгоритма Кока-Янгера-Касами и алгоритма Эрли. В качестве наиболее подходящего для синтаксического анализа открытых интерфейсных языков был выбран алгоритм Эрли.

Известно, что алгоритм Эрли производит только распознавание входной строки и не производит построение деревьев вывода. Поэтому необходимо было модифицировать этот алгоритм таким образом, чтобы он производил такое построение. Это было сделано в данной работе и модифицированный алгоритм был назван адаптированным для построения деревьев вывода алгоритмом Эрли. Наряду с алгоритмом построения деревьев вывода был также создан алгоритм обхода всех построенных деревьев сверху вниз и слева направо.

На основании проведенного выбора наиболее подходящего для синтаксического анализа открытых интерфейсных контекстно-свободных языков, была реализована система динамической компиляции. Данная система была реализована традиционным способом разделения процесса синтаксического анализа на лексический и синтаксический анализы. Каждый терминальный символ в синтаксическом анализаторе представляет собой регулярный язык, представленный в лексическом анализаторе специальным лексическим типом. Ввиду того, что множество лексических типов может динамически пополняться, необходимо было разработать как метод оптимального представления лексических типов в лексическом анализаторе, так и алгоритм лексического анализа, основанный на таком представлении. В качестве такого представления был выбран минимизированный детерминированный конечный автомат. Лексический тип изначально задается пользователем с помощью регулярного выражения. Затем, встроенный в лексический анализатор компилятор регулярных выражений производит синтаксический анализ данного регулярного выражения в недетерминированный конечный автомат, а затем, из данного недетерминированного конечного автомата в несколько этапов получается минимизированный детерминированный конечный автомат. Также, необходимо было продумать метод взаимодействия синтаксического анализатора с семантическим анализатором системы, в которой данный язык выступает в качестве интерфейса. В качестве результата синтаксический анализатор выдает множество деревьев разбора (или единственное дерево, если входная грамматика однозначная) входной строки, где узлы, обозначаемые терминальными символами грамматики, имеют единственный атрибут – строку символов, возвращенную лексическим анализатором.

Глава 1

Первая глава посвящена обсуждению вопросов, связанных с адаптированным для построения деревьев вывода алгоритмом Эрли. Как известно, оригинальный алгоритм Эрли не производит построения деревьев вывода входной строки, а отвечает только на вопрос, выводится или нет данная входная строка в данной грамматике. Поэтому, необходимо модифицировать данный алгоритм таким образом, чтобы он производил построение всех деревьев разбора в данной грамматике. Модифицированный алгоритм Эрли производит построение деревьев разбора снизу вверх и справа налево. Часто, необходимо произвести обход каждого из деревьев сверху вниз и слева направо. Поэтому, в данной главе представлен также и алгоритм обхода построенных в результате модифицированного алгоритма Эрли деревьев вывода входной строки сверху вниз и слева направо.

Прежде чем начать излагать произведенную модификацию алгоритма Эрли, необходимо напомнить в чем заключается оригинальный алгоритм Эрли. Алгоритм Эрли оперирует списками так называемых ситуаций Эрли. Каждая ситуация Эрли состоит из трех компонентов: помеченного правила грамматики (где помеченное правило это просто правило грамматики с точкой где-нибудь в правой части), номера списка, в котором начался вывод данного правила, и строки предосмотра, состоящей из терминальных символов грамматики. Известно, что использование предосмотров не дает сколько-нибудь заметного улучшения производительности, но влияет на количество ситуаций Эрли в списках ситуаций. Поэтому, использовать третий компонент ситуации Эрли мы не будем. Итак, ситуация Эрли это пара, состоящая из правила грамматики с точкой правой части и некоторого числа, большего нуля и меньшего, либо равного длине входной строки. Например, для правила грамматики вида $A \to X_1 ... X_k$ можно построить $k+1$ помеченных правил, начиная с $A \to \bullet X_1 ... X_k$, $A \to X_1 \bullet X_2 ... X_k$ и заканчивая $A \to X_1 ... X_k \bullet$. Второй компонент пары – это некоторое число i, $0 \le i \le |\omega|$, где $\omega = a_1 ... a_n$ - это входная строка, а $|\omega|$ - это ее длина. Ситуацию Эрли, состоящую из помеченного правила $A \to X_1 ... X_l \bullet X_{l+1} ... X_k$ и второго компонента i, будем обозначать как $[A \to X_1 ... X_l \bullet X_{l+1} ... X_k, i]$. Алгоритм Эрли состоит в последовательном

построении списков ситуаций Эрли (называемых состояниями Эрли). Список строится для каждого символа входной строки и для нулевой позиции при ее чтении.

Мы произведем модификацию алгоритма Эрли путем изменения определения ситуации Эрли, добавив в нее два дополнительных компонента: ссылку l на ту же ситуацию Эрли, что и текущая, но с точкой, левее на символ. Второй дополнительный компонент – это список ссылок $\langle p \rangle$ на ситуации, приведшие к передвижению точки на символ правее и добавлении текущей ситуации в данный список. Мы докажем в дальнейшем, что ссылка l содержит указатель на левый узел того же уровня в дереве вывода входной строки, а список ссылок $\langle p \rangle$ указатели на поддеревья нижнего уровня дерева вывода входной строки. Т.е., фактически, во время исполнения модифицированного алгоритма Эрли, производится построение всех деревьев вывода входной строки. Важным моментом здесь является то, что для сравнения ситуаций между собой мы будем использовать только первые три компонента: помеченное правило, номер списка и ссылку l, список ссылок $\langle p \rangle$ в сравнении не участвует. В противном случае, возможны случаи когда в каком-нибудь списке расширенных ситуаций Эрли может быть неограниченное число элементов и, следовательно, модифицированный алгоритм Эрли не сможет закончить свое исполнение.

Для доказательства корректности адаптированного для построения деревьев вывода алгоритма Эрли необходимо сначала показать, что количество ситуаций в каждом из списков ограничено сверху. Эту задачу выполняют следующие две леммы:

Лемма 1. Максимальное число ситуаций Эрли вида $\left[A \to X_1 \ldots X_k \bullet X_{k+1} \ldots X_m, j, l, \langle p \rangle\right]$ с одними и теми же первыми двумя компонентами $A \to X_1 \ldots X_k \bullet X_{k+1} \ldots X_m$ и j, отличающихся только значением ссылки l, принадлежащих состоянию $V[i]$, $j \le i \le |\omega|$, есть:

$S_{ij}^k = \dfrac{(i-j+k-1)!}{(i-j)!(k-1)!}$, если $1 \le k \le m$,

$S_{ij}^k = 1$, если $k = 0, i = j$,

$S_{ij}^k = 0$, если $k = 0, i \ne j$.

Если входная грамматика однозначная, то $S_{ij}^k = 1$ для всех $1 \le k \le m$.

Лемма 2. Максимальное количество ситуаций Эрли в состоянии Эрли $V[i]$ есть величина порядка $|P| \times (M_p + 1) \times (i+1) \times \dfrac{(M_P + i + 1)!}{(i+1)! M_P!}$, где M_P - максимальная из длин правых частей правил входной грамматики $G = \{T, N, P, S\}$. Если исходная грамматика однозначная, то максимальное количество ситуаций есть $|P| \times (M_p + 1) \times (i+1)$.

После того, как мы показали, что количество ситуаций Эрли в каждом состоянии Эрли ограничено сверху, необходимо доказать, что модифицированный алгоритм Эрли строит всевозможные деревья вывода входной строки даже и в том случае, если входная грамматика неоднозначная. Для этого нам необходима следующая

Лемма 3. Ситуация Эрли $\left[A \to \alpha \bullet \beta, j, l, \langle p \rangle\right]$ принадлежит состоянию Эрли $V[i]$ тогда и только тогда, когда $\alpha \Rightarrow^* a_{j+1} \ldots a_i$ и существуют такие строки γ и δ, что $S \Rightarrow^* \gamma A \delta$ и $\gamma \Rightarrow^* a_1 \ldots a_j$.

При этом, если строка α равна:

1. $\alpha' a$, где a - это терминальный символ входной грамматики. Тогда $a = a_i$ и $\alpha' \Rightarrow^* a_{j+1}...a_{i-1}$ и вывод имеет вид $S \Rightarrow^* \gamma A\delta \Rightarrow^* a_1...a_j A\delta \Rightarrow a_1...a_j\alpha\beta\delta \Rightarrow^* a_1...a_j\alpha' a\beta\delta$ $\Rightarrow^* a_1...a_{i-1}a\beta\delta$. Ситуация $\left[A \to \alpha \bullet \beta, j, l, \langle p \rangle\right]$ в этом случае есть $\left[A \to \alpha' a_i \bullet \beta, j, l, \varnothing\right]$, где $l = \left[A \to \alpha' \bullet a_i\beta, j, l', \langle p \rangle'\right] \in V[i-1]$.

2. $\alpha' B$, где B - это нетерминальный символ входной грамматики. Тогда существует число k, $j \le k \le i$ такое, что $\alpha' \Rightarrow^* a_{j+1}...a_k$ $X \Rightarrow^* a_{k+1}...a_i$ и вывод имеет вид $S \Rightarrow^* \gamma A\delta \Rightarrow^* a_1...a_j A\delta$ $\Rightarrow a_1...a_j\alpha\beta\delta \Rightarrow^* a_1...a_j\alpha' B\beta\delta \Rightarrow^* a_1...a_k B\beta\delta \Rightarrow^* a_1...a_i\beta\delta$. В этом случае, ситуация $\left[A \to \alpha \bullet \beta, j, l, \langle p \rangle\right]$ есть $\left[A \to \alpha' B \bullet \beta, j, l, \langle p \rangle\right]$, где $l = \left[A \to \alpha' \bullet B\beta, j, l', \langle p \rangle'\right] \in V[k]$ и список $\langle p \rangle$ содержит ссылку $p = \left[B \to \eta\bullet, k, l'', \langle p \rangle''\right] \in V[i]$.

3. Λ, где Λ - пустая строка, т.е. помеченное $A \to \alpha \bullet \beta$ правило есть $A \to \bullet\beta$, то ситуация $\left[A \to \alpha \bullet \beta, j, l, \langle p \rangle\right]$ имеет вид $\left[A \to \bullet\beta, j, null, \varnothing\right]$, $i = j$ и вывод имеет вид $S \Rightarrow^* \gamma A\delta \Rightarrow^* a_1...a_j A\delta \Rightarrow a_1...a_j\beta\delta$.

Таким образом, лемма 3 говорит о том, что каждая ситуация Эрли вида $\left[A \to \alpha \bullet \beta, j, l, \langle p \rangle\right]$ не просто так попадает в список, соответствующий некоторому символу входной строки a_i, а только, и только, если существует вывод $S \Rightarrow^* a_1...a_i\alpha\delta \Rightarrow^* a_1...a_j\delta$. Поэтому, если входная строка $\omega = a_1...a_n$ выводится во входной грамматике $G = \{T, N, P, S\}$, т.е. существует вывод $S \Rightarrow^+ a_1...a_n$, то в списке, соответствующем терминалу a_n должна присутствовать ситуация вида $\left[S \to X_1...X_k\bullet, 0, l, \langle p \rangle\right]$. Верно и обратное, а значит, верна

Теорема 1. Адаптированный для построения деревьев вывода алгоритм Эрли успешно заканчивает свою работу только, и только, если для входной строки ω существует хотя бы один вывод $S \Rightarrow^+ \omega$.

Во второй части первой главы рассматривается алгоритм обхода сверху вниз и слева направо всех деревьев вывода входной строки, построенных в результате исполнения адаптированного для построения деревьев вывода алгоритма Эрли. Такой алгоритм бывает необходим ввиду того, что алгоритм Эрли производит построение деревьев вывода снизу вверх и справа налево, что часто бывает неудобно для семантических анализаторов. Кроме того, алгоритм обхода производит построение всех деревьев вывода в явном виде. В адаптированном алгоритме Эрли узлы деревьев вывода скрыты внутри ситуаций Эрли как дополнительные компоненты l и $\langle p \rangle$.

Для доказательства корректности алгоритма необходимо показать, что алгоритм всегда заканчивает свою работу и что алгоритм производит обход всевозможных деревьев вывода входной строки. Для этого нам необходима лемма

Лемма 4. Алгоритм обхода деревьев, построенных в результате исполнения адаптированного алгоритма Эрли всегда заканчивает свою работу за конечное число шагов.

Следующая теорема завершает доказательство корректности работы алгоритма.

Теорема 2. Алгоритм обхода деревьев, построенных в результате исполнения адаптированного алгоритма Эрли, примененного к терминальной строке $\omega = a_1 ... a_n$ и КС-грамматике $G = \{N, T, P, S\}$, строит множество всех деревьев вывода данной терминальной строки $\omega = a_1 ... a_n$.

Глава 2

Во второй главе производится оценка вычислительной сложности как функции от объема входной грамматики, ее ветвистости и сложности, адаптированного для построения деревьев вывода алгоритма Эрли, алгоритма обхода деревьев вывода, построенных в результате исполнения адаптированного алгоритма Эрли и алгоритма Кока-Янгера-Касами. Чтобы оценить производительность алгоритма Кока-Янгера-Касами необходимо также рассмотреть производительность трех алгоритмов, позволяющих преобразовать входную грамматику в нормальную бинарную форму Хомского и алгоритма построения дерева вывода по таблице нетерминальных символов, построенных в результате исполнения алгоритма Кока-Янгера-Касами.

Адаптированный для построения деревьев вывода алгоритм Эрли во время своей работы производит манипуляции со списками ситуаций Эрли и сущностями входной грамматики. Входная грамматика, будучи представленной в памяти программы синтаксического анализатора, имеет естественное упорядочивание: по мере добавления память программы терминальных и нетерминальных символов грамматики, каждому из них дается порядковый номер, а правила грамматики могут быть представлены как векторы, в которых первым элементом выступает номер символа левой части правила, а остальные элементы представляют собой номера символов в правой части правила. Сами правила также могут быть естественным образом пронумерованы. Тогда, операции над сущностями входной грамматики сводятся к трем основным операциям: операции на вектором символов грамматики, операции над вектором правил грамматики и операции над вектором, представляющим данное правило грамматики. При оценке вычислительной сложности алгоритмов мы будем полагать элементарным каждое обращение к одном из указанных выше трех векторов. Алгоритмы семейства Эрли работают также с ситуациями Эрли, элементами вида $\left[A \to X_1 ... X_l \bullet X_{l+1} ... X_k, i, l, \langle p \rangle\right]$, где $A \to X_1 ... X_l \bullet X_{l+1} ... X_k$ - помеченное правило входной грамматики, i - номер одного из списков ситуаций Эрли, l - ссылка на некоторую ситуацию Эрли, а $\langle p \rangle$ - список таких ссылок. Помеченное правило в памяти программы представимо в виде пары чисел (r, t), где r - номер правила грамматики в представлении программы, а t - номер позиции точки в правиле. Ссылку на ситуацию легко представить как пару номеров, где первый элемент – это номер списка ситуаций, а второй – порядковый номер ситуации Эрли в этом списке. Таким образом, ссылка l может быть представлена парой чисел, а список ссылок $\langle p \rangle$ - вектором таких пар. Отсюда видно, что, например, операция передвижения точки в помеченном правиле, принадлежащем некоторой ситуации, на символ правее, сводится просто к увеличению на единицу числа t в паре (r, t), представляющем данное помеченное правило. Сравнение текущего символа входной строки с символом после точки в помеченном правиле, в конечном итоге, сводится к обращению к элементу вектора символов грамматики и сравнению его значения с данным символом. Таким же образом можно элементарным образом организовать все операции с ситуацией Эрли. Единственные сомнения вызывает список $\langle p \rangle$. Но, как это было показано в первой главе, количество ситуаций в списке ограничено сверху. Поэтому, хранение элементов в списке $\langle p \rangle$ можно организовать двояко: в виде списка для последовательного прохождения и

в виде вектора длиной совпадающего с максимальным числом ситуаций в предыдущих списках. Тогда, операция проверки присутствия данной ссылки в списке $\langle p \rangle$ сводится к вычислению индекса данной ситуации в векторе, соответствующем данному списку. Итак, все операции с ситуациями Эрли мы будем считать элементарными. Поэтому, вычислительная сложность алгоритмов семейства Эрли будет оцениваться в терминах количества ситуаций в списке и частоты обращения с сущностями грамматики.

Для оценки вычислительной сложности адаптированного для построения деревьев вывода алгоритмов Эрли нам необходимы сначала следующие леммы:

Лемма 5. Для любой ситуации Эрли $[A \to \alpha \bullet \beta, k, l, \langle p \rangle]$, принадлежащей состоянию $V[i]$, максимальное количество элементов в списке $\langle p \rangle$ есть $F_P \times \frac{(M_P + i + 1)!}{(i+1)! M_P!}$, если входная грамматика неоднозначная, и 1 если входная грамматика однозначная.

Лемма 6. Операции поиска, добавления и удаления любой ситуации Эрли вида $[A \to \alpha \bullet \beta, k, l, \langle p \rangle]$ в любом состоянии Эрли $V[i]$ занимают постоянное время.

Следующая теорема дает оценку вычислительной сложности адаптированного для построения деревьев вывода алгоритма Эрли в терминах объема входной грамматики, ее ветвистости и протяженности.

Теорема 3. Максимальное время выполнения адаптированного для построения деревьев вывод алгоритма Эрли есть функция порядка $O\left(F_P \times |P|^2 \times M_P^2\right)$, если входная грамматика однозначная и $O\left(F_P \times |P|^2 \times M_P^{|\omega|+2}\right)$ если входная грамматика неоднозначная.

Как видно из теоремы, вычислительная сложность алгоритма для неоднозначной грамматики увеличивается на величину $M_P^{|\omega|}$, т.е при неоднозначных грамматиках невыгодно иметь правила большой длины, лучше заменять их большим количеством коротких правил.

Оценка вычислительной сложности алгоритма обхода сверху вниз и слева направо построенных в результате исполнения адаптированного алгоритма Эрли деревьев, дается в следующей

Теорема 4. Максимальное время выполнения алгоритма 2 есть функция $O\left(|P| \times M_P\right)$ для однозначных грамматик и функция $O\left(|P| \times F_P^2 \times M_P^{3|\omega|+4}\right)$ для неоднозначных.

Приступим теперь к оценке вычислительной сложности алгоритмов, связанных с алгоритмом Кока-Янгера-Касами. Как известно, данный алгоритм работает только с грамматиками, удовлетворяющими нормальной бинарной форме Хомского. Поэтому, перед тем как приступить к оценке, собственно, алгоритма Кока-Янгера-Касами, мы должны рассмотреть алгоритм преобразования входной грамматики в нормальную бинарную форму Хомского. На самом деле, для преобразования входной грамматики в нормальную бинарную форму нам необходимы три алгоритма: алгоритм определения нетерминалов грамматики, порождающих пустую строку, алгоритм преобразования входной грамматики в эквивалентную ей грамматику без правил с пустой правой частью и, собственно, алгоритм преобразования грамматики без правил с пустой правой частью в грамматику, удовлетворяющую нормальной бинарной форме.

Для оценки алгоритма определения порождающих пустую строку нетерминалов грамматики нам необходима следующая

Лемма 7. Пусть нетерминал $A \in N$ порождает пустую строку и минимальная высота дерева вывода этого нетерминала в пустую строку равна H. Тогда нетерминал A будет добавлен в N_ε на H-ом повторении шага 1 алгоритма определения порождающих пустую строку нетерминалов.

Следующая теорема дает оценку вычислительной сложности алгоритма определения порождающих пустую строку нетерминалов как функции от объема грамматики.

Теорема 5. Алгоритм определения порождающих пустую строку нетерминалов выполняется за время $O\left(|G|^2\right)$.

Оценим теперь вычислительную сложность алгоритма преобразования грамматики в эквивалентную ей грамматику, не содержащую правил с пустой правой частью. Здесь мы должны также оценить объем полученной в результате преобразования грамматики.

Теорема 6. Алгоритм преобразования грамматики в эквивалентную ей грамматику, не содержащую правил с пустой правой частью выполняется за время $O\left(|G|\right)$ и объем грамматики G', полученной в результате исполнения алгоритма данного алгоритма, также равен $O\left(|G|\right)$.

Оценка вычислительной сложности алгоритма преобразования грамматики без правил с пустой правой частью в эквивалентную ей грамматику в нормальной форме Хомского, а также оценка объема полученной в результате исполнения грамматики, дается в следующей теореме.

Теорема 7. Алгоритм преобразования грамматики без правил с пустой правой частью в эквивалентную ей грамматику в нормальной форме Хомского выполняется за время $O\left(|G|\right)$. Объем выходной грамматики G' есть также величина $O\left(|G|\right)$.

Итак, мы оценили вычислительную сложность алгоритмов преобразования исходной грамматики в нормальную форму Хомского. Теперь необходимо оценить вычислительную сложность алгоритма Кока-Янгера-Касами. Как известно, алгоритм Кока-Янгера-Касами не строит дерева вывода входной строки, поэтому нам необходимо также оценить вычислительную сложность алгоритма построения дерева вывода на основе таблицы нетерминалов, построенной в результате исполнения алгоритма Кока-Янгера-Касами. Итак, вычислительная сложность алгоритма Кока-Янгера-Касами оценивается в следующей теореме.

Теорема 8. Алгоритм Кока-Янгера-Касами выполняется за максимальное время $O\left(|G|^2\right)$.

Алгоритм построения дерева вывода по таблице нетерминальных символов, построенной в результате исполнения алгоритма Кока-Янгера-Касами, имеет следующую вычислительную сложность.

Теорема 9. Максимальное время работы алгоритма построения дерева вывода по таблице нетерминальных символов, построенной в результате исполнения алгоритма Кока-Янгера-Касами есть $O\left(|G|^2\right)$.

Итак, мы оценили два множества алгоритмов: два алгоритма, связанных с алгоритмом Эрли, и пять алгоритмов, связанных с алгоритмом Кока-Янгера-Касами. На основании этой оценки можно утверждать, что наиболее подходящим для наших целей построения синтаксического анализатора для открытых интерфейсных языков, является алгоритм Эрли.

Действительно, алгоритмы Эрли и алгоритмы Кока-Янгера-Касами имеют приблизительно одинаковую вычислительную сложность от объема грамматики, но алгоритм Эрли не требует преобразования входной грамматики в нормальную бинарную форму. Следовательно, для реализации мы выбираем адаптированный для построения деревьев вывода алгоритм Эрли.

Глава 3

Третья глава посвящена описанию проблем, связанных с построением динамической системы компиляции с использованием адаптированного для построения деревьев вывода алгоритма Эрли. Система динамической компиляции реализована согласно классической модели, при которой процесс синтаксического анализа, с целью упрощения восприятия грамматики языка человеком, разделяется на две фазы: фазу лексического анализа и, собственно, фазу синтаксического анализа. Модуль лексического анализатора обрабатывает входной текст и выдает в результате поток слов-лексем, каждое из которых принадлежит определенному лексическому типу. Каждый лексический тип представлен в модуле синтаксического анализатора как соответствующий терминальный символ входной грамматики. Таким образом осуществляется связь между модулями синтаксического и лексического анализаторов. Также, необходимо рассмотреть вопросы, связанные с реализацией взаимодействия между синтаксическим и семантическим анализаторами. Существуют несколько различных способов организации такого взаимодействия, нам необходимо выбрать наиболее приемлемый для анализа открытых интерфейсных языков.

Интерфейс модуля синтаксического анализатора совмещает как функции, непосредственно относящиеся к синтаксическому анализатору, так и функции, позволяющие проводить модификацию входной грамматики. Группа, состоящая из четырех методов: AddTerminal, AddNonterminal, AddRule и SetStartSymbol позволяет проводить модификацию грамматики синтаксического анализатора. Вторая группа методов относится, собственно. к реализации синтаксического анализатора. Из них основным является метод Run, позволяющий производить запуск алгоритма синтаксического анализа на основе добавленных ранее сущностей входной грамматики.

Для удобной и надежной организации работы модуля синтаксического анализатора необходимо его разбить, по крайней мере, на два подмодуля: подмодуль грамматики и подмодуль синтаксического анализатора. Подмодуль грамматики осуществляет хранение сущностей грамматики: нетерминальных и терминальных символов и правил грамматики. Как уже указывалось, при добавлении новой сущности в модуль грамматики ей дается уникальный номер, который, в дальнейшем, участвует во всех операциях над элементами грамматики, так или иначе, связанных с данной сущностью. Множества терминальных и нетерминальных символов грамматики представляется, соответственно, двумя векторами. Правила грамматики хранятся в виде матрицы вектора векторов, каждый элемент большого вектора представляет собой вектор символов грамматики, первый элемент которого есть символ левой части правила, а остальные – символы правой части. Модуль позволяет добавлять новые сущности грамматики, устанавливать начальный символ грамматики и проводить различные операции поиска по элементам грамматики. Второй подмодуль представляет собой реализацию синтаксического анализатора. Главным методом модуля является метод запуска алгоритма синтаксического анализа. Во время исполнения алгоритма синтаксического анализа модуль взаимодействует с модулем грамматики с целью осуществления операций поиска сущностей грамматики, последовательного прохода по элементам грамматики и сравнения символов грамматики с различными операндами алгоритма синтаксического анализа. Алгоритм синтаксического анализа производит построение списков ситуаций Эрли. Каждая ситуация Эрли представляет собой четверку, в которой первый элемент – помеченное правило, есть пара неотрицательных целых: номера правила грамматики, данного модулем грамматики, и номера позиции точки в правой части правила. Второй элемент – неотрицательное целое, представляющее собой номер некоторого

списка ситуаций Эрли. Третий элемент – ссылка на некоторую, уже существующую, ситуацию Эрли, представляется парой неотрицательных целых, в которое в качестве первого элемента выступает номер списка ситуаций Эрли, а второго – порядковый номер ситуации в списке. Четвертый элемент – список ссылок на ситуации Эрли, представляется вектором пар неотрицательных целых.

Лексический анализатор реализован в виде отдельного модуля, предоставляющего интерфейс для добавления новых лексических типов и получения из входного текста очередной лексемы., Метод, позволяющий получить очередную лексему от лексического анализатора, традиционно называется GetToken. Обычный сценарий работы с лексическим анализатором строится по следующей схеме: сначала, с помощью метода AddLexType, в лексический анализатор добавляется множество лексических типов. Затем вызывается метод Initialize который инициализирует модуль лексического анализатора входным текстом. Далее, вызывается метод GetToken для взятия очередной лексемы из входного текста.

При реализации лексического анализатора в виде динамически пополняющейся коллекции лексических типов, необходимо разработать метод представления регулярного языка лексического типа удобный как для человека, так и для представления в памяти программы. Ввиду того, что язык лексического типа является регулярным, удобным для человеческого восприятия кажется определение данного языка в виде регулярного выражения. В связи с этим, как часть модуля лексического анализатора, был разработан компилятор языка регулярных выражений. Синтаксис данного языка представляет один из диалектов языка регулярных выражений для языка программирования Perl. При добавлении нового лексического типа в программу лексического анализатора регулярный язык данного типа описывается в виде регулярного выражения, передаваемого в процедуру добавления данного лексического типа. В качестве результата процедура добавления возвращает неотрицательное целое, представляющее собой идентификатор добавленного лексического типа. Этот идентификатор будет потом возвращен в процессе исполнения алгоритма синтаксического анализа методом GetToken, если во входном тесте была распознана лексема, принадлежащая языку данного лексического типа.

Регулярное выражение, описывающее язык нового лексического типа, разбирается в соответствии с синтаксисом языка регулярных выражений и компилируется в недетерминированный конечный автомат, задающий тот же регулярный язык, что и данное регулярное выражение. Затем, из недетерминированного конечного автомата методом построения подмножеств строится эквивалентный ему детерминированный конечный автомат. Затем этот автомат минимизируется по числу состояний и, в результате, данный лексический тип представляется в программе минимизированным детерминированным конечным автоматом, задающим тот же язык, что и переданное в качестве параметра в процедуру добавления лексического типа, регулярное выражение. Задание в программе регулярного языка лексического типа в виде минимизированного детерминированного конечного автомата очень удобно для последующих манипуляций, производимых в процессе лексического анализа.

На основе множества минимизированных детерминированных автоматов, принадлежащих соответствующим лексическим типам, производится лексический анализ входного текста. Главной особенностью здесь является то, что множество конечных автоматов лексических типов задается динамически между двумя исполнениями алгоритма синтаксического анализа. Поэтому, построить из этого множества детерминированный автомат, как это обычно делается в генераторах лексических анализаторов, не представляется возможным. Поэтому, при каждом запуске процедуры выделения лексемы из входного текста, производится моделирование детерминированного конечного автомата из недетерминированного, сформированного на основе множества минимизированных детерминированных конечных автоматов лексических типов.

Необходимо было также продумать вопрос взаимодействия между модулем синтаксического анализатора и внешней программой семантического анализатора. Такое

взаимодействие можно организовать несколькими способами, основных из два: вовлекать семантический анализатор во время исполнения алгоритма синтаксического анализа и вовлекать семантический анализатор после окончания работы алгоритма синтаксического анализатора. Для реализации был выбран второй подход. Модуль синтаксического анализатора во время исполнения алгоритма синтаксического анализа производит построение дерева вывода входной строки, т.е. строки, составленной из терминальных символов грамматики, возвращенных модулем лексического анализатора. Деревьев вывода может быть несколько, если входная грамматика неоднозначная. Каждый узел дерева вывода, помеченный терминальным символом, имеет единственный атрибут, представляющий подстроку входного текста, распознанную лексическим анализатором при возвращении данного терминального символа как лексемы, принадлежащей соответствующему лексическому типу.

Заключение

Основной целью работы являлось исследование особенностей синтаксического анализа открытых интерфейсных контекстно-свободных языков. Данную цель можно считать полностью реализованной. Все задачи, поставленные в работе, были выполнены. В качестве главного критерия эффективности работы алгоритма синтаксического анализа для открытого интерфейсного контекстно-свободного языка был предложен метод оценки вычислительной сложности алгоритма как функции от таких параметров входной контекстно-свободной грамматики, как ее объем, ветвистость и протяженность. На основании этого метода, была произведена оценка эффективности семейства алгоритмов Кока-Янгера-Касами и семейства алгоритмов Эрли. В работе был введен новый алгоритм – адаптированный для построения деревьев вывода алгоритм Эрли, и проведено доказательство корректности работы данного алгоритма. Введенный алгоритм также может применяться для синтаксического анализа контекстно-свободных языков, грамматики которых не удовлетворяют каким-либо ограничениям. В частности, адаптированный алгоритм Эрли позволяет производить построение деревьев вывода для языков, заданных неоднозначными грамматиками. Проблема построения всех деревьев вывода для неоднозначных контекстно-свободных языков приобрела особенную актуальность в последнее десятилетие и была решена в общем виде совсем недавно [10]. Однако, подходящего алгоритма синтаксического анализа, позволяющего производить построение всех деревьев вывода анализируемой строки для открытых языков, до настоящего времени не существовало. Адаптированный алгоритм Эрли дает возможность производить такое построение. В данной работе были выделены такие свойства контекстно-свободной грамматики, как объем, ветвистость и протяженность. Представляется интересным исследовать не сохраняются ли эти свойства при естественных преобразованиях грамматик, таких как преобразование исходной грамматики в нормальную бинарную форму или в форму Грейбах [1], и попытаться выделить какие-нибудь инварианты таких преобразований на основе полученных в работе свойств контекстно-свободной грамматики.

Список литературы

1. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции, Том 1. – Москва: Мир, 1978. – 612 с.
2. Бениаминов Е. М., Болдина Д. М. Система представления и обработки знаний ЭЗОП//Материалы конференции Диалог-2001: Прикладные проблемы. – 2001. – С. 41-43.
3. Бениаминов Е. М., Вайнтроб А. Ю. Основные принципы диалогового языка для представления знаний средствами категорного подхода//Материалы конференции ДИАЛОГ-87. – Тбилиси, 1988. – С. 174-177.
4. Бениаминов Е. М., Манушина М. Ю. Принципы построения открытого языка шаблонных выражений в системе представления знаний//НТИ Сер. 2. – 2000. – № 7.
5. Earley J. An efficient context-free parsing algorithm//ACM. – 13. – 2. – p. 94-102.
6. Earley J. An efficient context-free parsing algorithm: Ph. D. Dissertation. – Pittsburg: Carnegie Mellon University, 1968.
7. Graham S., Harrison M., Russo W. An improved Context-Free Recognizer//ACM TOPLAS. – 1980: vol. 2. – № 3. – p. 415-462.
8. Kasami T., Torii K. A syntax-analysis procedure for unambiguous context-free grammars//ACM 16. – 1969. – № 3. – p. 423-431.
9. Knuth D. E. On the translation of languages from left to right//Inform. Control 8. – 1965. – p. 607-639.
10. Scott E., Johnstone A., Hussain S. S. Tomita-Style Generalized LR Parsers//Royal Holloway University of London. – 2000.
11. Tomita M. An Efficient Augmented-Context-Free Parsing Algorithm. – Pittsburgh: Carnegie-Mellon University. – 1987.
12. Tomita M. Efficient parsing for natural language. – Boston: Kluwer Academic Publishers, 1986. – p. 201.
13. Younger D. H. Recognition of context-free languages in time n^3//Inform. Control 10. – 1967. № 2. – p. 189-208.

Основные результаты диссертации изложены в следующих публикациях:

1. Лапшин В. А. Оценка производительности алгоритма Кока-Янгера-Касами как функции от объема грамматики//НТИ Сер. 2. – 2004. – № 9. – С. 9-15.
2. Лапшин В. А. Особенности синтаксического анализа открытых интерфейсных контекстно-свободных языков//НТИ Сер. 2. – 2004. – № 12. – С. 29-33.
3. Лапшин В. А. Оценка вычислительной сложности адаптированного для построения деревьев вывода алгоритма Эрли//НТИ Сер. 2. – 2005. – № 4. – С. 1-14.
4. Лапшин В. А. Адаптированный для построения деревьев вывода алгоритм Эрли//НТИ Сер. 2. – 2005. – № 5. – С. 1-12.