МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Ижевский государственный технический университет имени М.Т.Калашникова"
(ФГБОУ ВПО «ИжГТУ имени М.Т.Калашникова»)

Кучуганов В.Н., Касимов Д.Р.

МАТЕМАТИЧЕСКАЯ ЛИНГВИСТИКА
МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №5
«РАЗРАБОТКА ГЕНЕРАТОРА»

Рекомендовано учебно-методическим советом ФГБОУ ВПО «ИжГТУ имени М.Т. Калашникова» для использования в учебном процессе в качестве
элемента ЭУМКД для студентов обучающихся по направлению 230100.62 «Информатика и вычислительная техника», профилям «Автоматизированные системы обработки информации и управления», «Системы автоматизированного проектирования» при изучении дисциплин «Математическая лингвистика», «Лингвистическое обеспечение САПР»

Ижевск 2013

Составители: Кучуганов Валерий Никонорович, доктор технических наук, профессор
Касимов Денис Рашидович, ассистент

УДК 681.3

Математическая лингвистика: методические указания к выполнению лабораторной работы №5 «Разработка генератора» по курсам «Математическая лингвистика», «Лингвистическое обеспечение САПР» профилей «Автоматизированные системы обработки информации и управления», «Системы автоматизированного проектирования» направления 230100.62 «Информатика и вычислительная техника».
Составители: Кучуганов В.Н., Касимов Д.Р., Ижевский государственный технический университет имени М.Т. Калашникова. Ижевск, 2013. – 12 с.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

До сих пор мы рассматривали процесс синтаксического анализа только как процесс анализа допустимости входной цепочки. В лингвистическом процессоре синтаксический анализ служит основой еще одного важного шага – построения дерева синтаксического разбора. Построение дерева синтаксического разбора является простейшим частным случаем трансляции – процесса преобразования некоторой входной цепочки в некоторую выходную.

Трансляция представляет собой отображение входного потока информации в выходной. Пусть Т – входной алфавит, а П – выходной алфавит. Переводом (или трансляцией) с языка $L1 \subseteq T^*$ на язык $L2 \subseteq П^*$ называется отображение τ: $L1 \rightarrow L2$. Если $y = \tau(x)$, то цепочка y называется выходом для цепочки x.

Существует несколько формализмов для определения переводов: схемы синтаксически управляемого перевода, атрибутные транслирующие грамматики, преобразователи с магазинной памятью.

В описываемой ниже лабораторной работе изучаются вопросы трансляции языков, управляемой контекстно-свободной грамматикой. Мы связываем информацию с конструкциями языка программирования с помощью атрибутов грамматических символов, представляющих данную конструкцию. Значения атрибутов вычисляются согласно «семантическим правилам», связанным с продукциями грамматики.

Концептуально мы разбираем входной поток токенов, строим дерево разбора и обходим его так, как необходимо для выполнения семантических правил в узлах дерева разбора. Выполнение семантических правил может генерировать код, сохранять информацию в таблице символов, выводить сообщения об ошибках или выполнять какие-либо другие действия. Результат трансляции потока токенов будет получен путем выполнения указанных семантических правил.

Реализация не всегда следует описанной схеме. Трансляция может быть реализована за один проход выполнением семантических правил в процессе синтаксического анализа, без явного построения дерева разбора. Однопроходная реализация важна с точки зрения скорости трансляции.

Цель лабораторной работы – ознакомиться с теоретическими и практическими основами построения блока синтеза лингвистического процессора.

1. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ №5

Тема работы: «Разработка генератора».

Цель работы: ознакомиться с теоретическими и практическими основами построения блока синтеза лингвистического процессора.

Используемые программные средства: система программирования Delphi 7.0 или выше; графический редактор Microsoft Visio.

Задание по лабораторной работе. По результатам анализа исходного текста получить выходной текст в виде:

1. Все числа исходного текста должны быть переведены в десятичное представление.

2. Выполнить вывод полученного текста в структурированном виде с помощью отступов («ступеньками»).

Генератор должен осуществлять отдельный просмотр текста (синтаксического дерева). Правила структурирования текста разработать самостоятельно.

Содержание отчета:

1) титульный лист;

2) текст задания, включающий вариант задания;

3) атрибутная грамматика с атрибутами и действиями по преобразованию исходного текста в выходной;

4) исходный текст генератора;

5) результаты тестирования.

Методические рекомендации к лабораторной работе

В данной лабораторной работе все трансляции могут быть реализованы применением семантических правил вычисления атрибутов в дереве разбора по определенному порядку. Обход дерева начинается с корня и состоит в посещении каждого узла дерева в определенном порядке. Семантические правила применяются с использованием рекурсивного обхода, описанного ниже. Он начинается в корне дерева и рекурсивно проходит в порядке слева направо по всем дочерним узлам данного узла, как показано на рисунке 1.1. Семантические правила в данном узле применяются после посещения всех его потомков. Этот обход имеет также название «сперва вглубь», или «в глубину», поскольку в первую очередь посещаются еще не пройденные дочерние узлы.

Алгоритм рекурсивного обхода:

```
procedure visit(n: node);
begin
   for каждый дочерний узел m узла n в порядке слева направо do
      visit(m);
   применить семантические правила в узле n
end
```

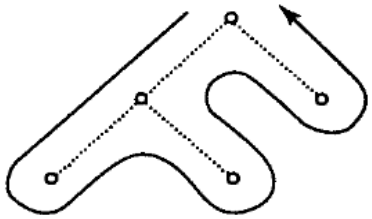

Рисунок 1.1. Пример рекурсивного обхода дерева “в глубину”

Варианты индивидуальных заданий

Данная лабораторная работа выполняется на основе результатов выполнения лабораторной работы № 4, и должна соответствовать ее варианту. Правила структурирования текста должны быть адекватны варианту грамматики.

Контрольные вопросы:

1. Конструирование генераторов.

2. Внутренние формы данных. Базовые данные. Массивы. Структуры. Линейные списки. Деревья. Графы.

3. Классификация и принципы построения команд компьютера.

4. Внутренние формы операторов. Перевод в трех-, двух-, одноадресную систему команд.

5. Внутренние формы операторов. Перевод в нульадресную (стековую) систему команд.

6. Методы распределения памяти.

7. Конструирование оптимизаторов.

8. Документирование перевода.

2. СХЕМЫ СИНТАКСИЧЕСКИ УПРАВЛЯЕМОГО ПЕРЕВОДА

Одним из формализмов, используемых для определения переводов, является схема синтаксически управляемого перевода. Фактически, такая схема представляет собой КС-грамматику, в которой к каждому правилу добавлен элемент перевода. Всякий раз, когда правило участвует в выводе входной цепочки, с помощью элемента перевода вычисляется часть выходной цепочки, соответствующая части входной цепочки, порожденной этим правилом.

Пусть задана следующая грамматика арифметических выражений:

$Z \rightarrow E$

$E \rightarrow T \mid E + T \mid E - T \mid - T$

$T \rightarrow F \mid T * F \mid T / F$

$F \rightarrow a \mid (E)$

Схема перевода, отображающего арифметические выражения в соответствующие постфиксные польские записи, представлена в таблице 2.1.

Таблица 2.1. Схема перевода инфиксной формы в польскую

№	Правило	Семантическая программа
1	$Z \rightarrow E$	
2	$E \rightarrow T$	
3	$E \rightarrow E + T$	Push(‘+’)
4	$E \rightarrow E - T$	Push(‘–’)
5	$E \rightarrow - T$	Push(‘@’)
6	$T \rightarrow F$	
7	$T \rightarrow T * F$	Push(‘*’)
8	$T \rightarrow T / F$	Push(‘/’)
9	$F \rightarrow a$	Push(a)
10	$F \rightarrow (E)$	

Здесь семантическая процедура Push(X) добавляет в конец выходной цепочки символ X.

Для того чтобы реализовать алгоритм разбора без полного перебора возможных вариантов применимости правил, нам потребуется стек S и переменная R, которая будет хранить очередной считываемый символ.

При этом надо сделать следующие замечания: правило (1) применимо, если R=’$’ (символ $ означает конец анализируемой последовательности); правила (2), (3), (4) применимы, если R равен '+', '-', '$' или ')'.

Алгоритм синтаксически управляемого перевода выглядит так: сначала в стек S заносится символ $. Далее к содержимому стека мы пробуем применить какое-либо правило из списка. Если ни одно из правил не срабатывает, то в стек заносится очередной символ анализируемой входной последовательности. Проще всего изобразить процедуру разбора на конкретном примере. В таблице 2.2 представлен разбор выражения "a*(b+c)$" (в столбце ωk... записан остаток входной цепочки символов).

Таблица 2.2. Разбор выражения "a*(b+c)$"

Стек S	R	ω_k...	Номер правила	Польская цепочка
$	a	*(b+c)$		
$a	*	(b+c)$	9	a
$F	*	(b+c)$	6	a
$T	*	(b+c)$		a
$T*	(	b+c)$		a
$T*(	b	+c)$		a
$T*(b	+	c)$	9	ab
$T*(F	+	c)$	6	ab
$T*(T	+	c)$	2	ab
$T*(E	+	c)$		ab
$T*(E+	c	)$		ab
$T*(E+c	)	$	9	abc
$T*(E+F	)	$	6	abc
$T*(E+T	)	$	3	abc+
$T*(E	)	$		abc+
$T*(E)	$		10	abc+

$T*F	$		7	abc+*
$T	$		2	abc+*
$E	$		1	abc+*
$Z	$		STOP	abc+*

Признак нормального окончания работы алгоритма: когда в стеке остался единственный символ Z, а текущим символом является ‘$’- символ конца входной последовательности, то мы считаем, что процедура синтаксического анализа завершена успешно. В противном случае (если в стеке есть другие символы) фраза построена неверно.

Основной недостаток синтаксически-управляемого перевода (как, впрочем, и всех механизмов, основанных на применении грамматик в явном виде) заключается в том, что фактически мы имеем дело с полным перебором всех возможных вариантов применений правил грамматики. Избежать этого перебора позволяют лишь введенные весьма искусственные соглашения относительно условий применимости тех или иных правил в различных ситуациях (см. те же правила (1), (2), (3) и (4)). Более того, поиск как таковой в схеме синтаксически управляемого перевода категорически недопустим.

3. АТРИБУТНЫЕ ТРАНСЛИРУЮЩИЕ ГРАММАТИКИ

Рассматриваемые в настоящем разделе атрибутные транслирующие грамматики отличаются от схем синтаксически управляемого перевода тем, что символам грамматики приписываются атрибуты, отражающие семантическую информацию, а правилам грамматики сопоставляются правила вычисления значений атрибутов.

Грамматику называют атрибутной грамматикой, если:

1. Символам грамматики приписаны один или несколько атрибутов и для каждого атрибута определено множество допустимых значений.

2. Атрибуты могут быть наследуемыми и синтезируемыми.

3. Для каждого правила грамматики должны быть заданы правила вычисления атрибутов в виде оператора присваивания с функцией в правой части, определяющей значение атрибута, расположенного слева. Такие функции для вычисления атрибутов могут зависеть от атрибутов правой или левой частей рассматриваемого правила.

4. Для наследуемых атрибутов начального символа должны быть заданы начальные значения.

5. Функции, вычисляющие значения синтезируемых атрибутов символов действия, должны зависеть от других атрибутов этого символа.

В качестве примера рассмотрим атрибутную грамматику, описывающую трансляцию арифметических выражений, состоящих из констант, в значение заданного выражения (таблица 3.1).

Таблица 3.1. Пример атрибутной транслирующей грамматики

Продукция	Семантические правила
$S \rightarrow E$	Ответ.↓val := E.↑val
$E \rightarrow E + T$	E.↑val := E.↑val + T.↑val
$E \rightarrow T$	E.↑val := T.↑val
$T \rightarrow T * P$	T.↑val := T.↑val * P.↑val
$T \rightarrow P$	T.↑val := P.↑val
$P \rightarrow (E)$	P.↑val := E.↑val
$P \rightarrow c$	P.↑val := c.↑val

У каждого нетерминала E, T, P имеется по одному атрибуту, принимающему целочисленные значения. Терминальный символ *c* также имеет один атрибут, определяющий значение константы и принимающий целочисленные значения. Операционный символ грамматики *Ответ* имеет наследуемый атрибут с целочисленной областью значений. Начальным символом грамматики служит символ S.

Атрибутная грамматика не задает конкретный порядок вычисления атрибутов в дереве разбора. Пригоден любой порядок, который определяет атрибуты *a* после всех атрибутов, от которых зависит *a*. Вообще говоря, некоторые атрибуты нужно вычислять при первом же посещении узла, а какие-то – во время или после прохода по дочерним узлам.

4. ПРИМЕР ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

На рисунке 4.1 приведен пример работы программы.

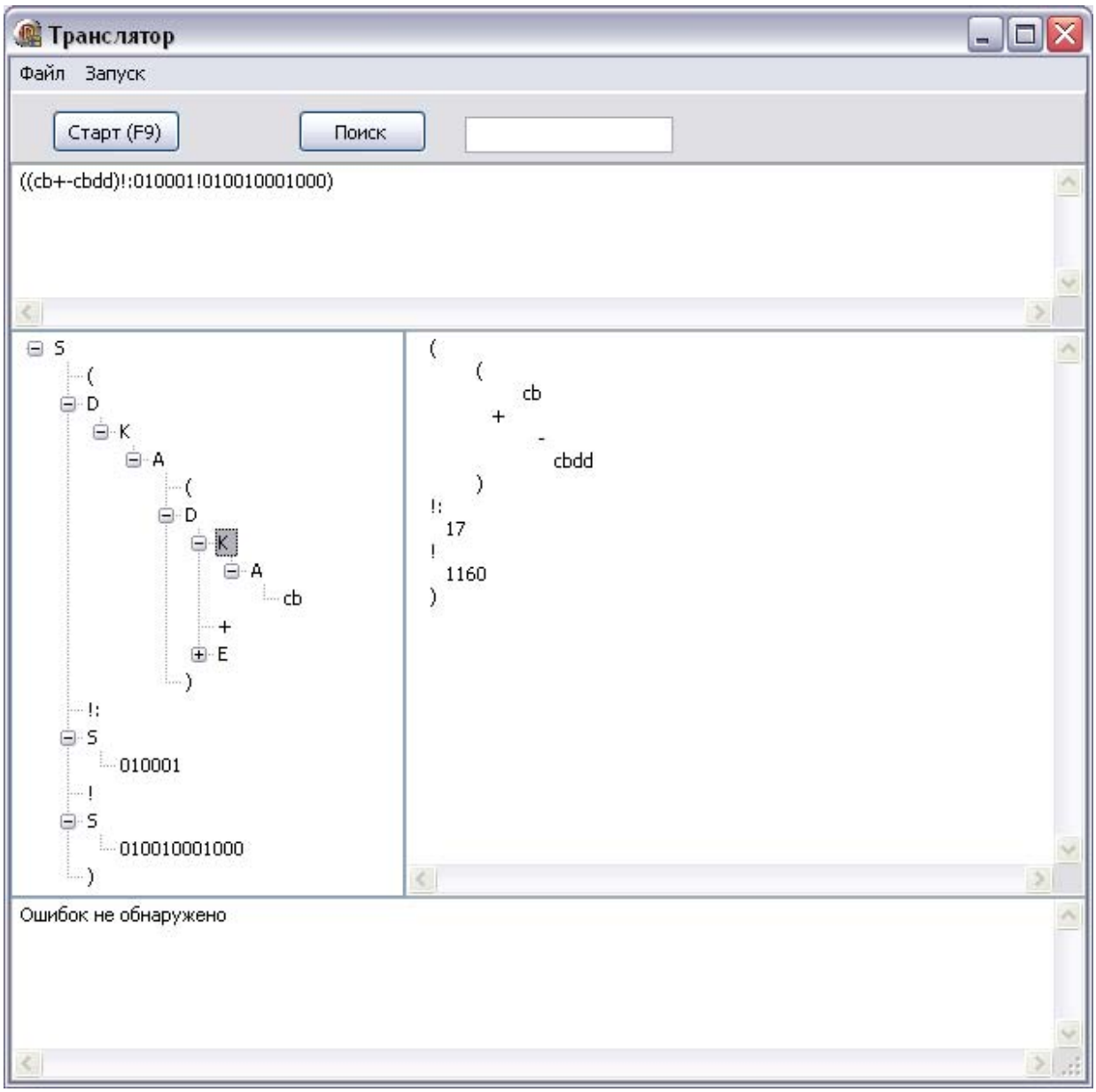

Рисунок 4.1. Пример работы программы

ЛИТЕРАТУРА

1. Ахо А., Лам М., Сети Р., Ульман Дж. Компиляторы: принципы, технологии и инструментарий, 2-е изд. : Пер. с англ. – М. : ООО "И.Д. Вильямс", 2008. – 1184 с. : ил.
2. Knuth D.E. Examples of formal semantics // Lecture Notes in Mathematics. - N.Y., Springer-Verlag, 1971. -V. 188. – P. 212-235.
3. И.Г. Кревский, М.Н. Селиверстов, К.В. Григорьева. Формальные языки, грамматики и основы построения трансляторов. Учебное пособие (под ред. д.т.н., профессора А.М. Бершадского). – Пенза, 2003.
4. Свердлов С. З. Языки программирования и методы трансляции: Учебное пособие. — СПб.: Питер, 2007. — 638 с: ил.
5. Льюис Ф., Розенкранц Д., Стирнз Р. «Теоретические основы проектирования компиляторов». – М.: Мир, 1979.
6. Компаниец Р. И., Маньяков Е. В., Филатов Н. Е., «Системное программирование. Основы построения трансляторов». – СПб.: КОРОНА принт, 2000.
7. Мозговой М. В. «Классика программирования: алгоритмы, языки, автоматы, компиляторы. Практический подход». – СПб.: Наука и Техника, 2006.