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 №4
«РАЗРАБОТКА КОНТЕКСТНОГО АНАЛИЗАТОРА»

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 №4 «Разработка контекстного анализатора» 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. – 11 с.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Спецификация языка часто содержит некоторые контекстные условия, которые не выражаются контекстно-свободными грамматиками. В такой ситуации кажется естественным обратиться к контекстно-зависимым грамматикам, однако эти грамматики в общем случае не применимы для определения контекстных условий практических языков.

Работоспособным подходом являются атрибутные грамматики, которые могут выполнять несколько полезных функций при определении синтаксиса и семантики языка. Атрибутная грамматика может быть использована для спецификации контекстно-зависимых аспектов синтаксиса языка, таких как проверка, что элемент был объявлен и что использование элемента согласуется с его объявлением.

Типичный лингвистический процессор проверяет контекстные условия на фазе семантического (или контекстного) анализа, следующей за (или проходящей одновременно с) фазой синтаксического анализа.

В данной лабораторной работе предлагается разработать атрибутную грамматику и парсер для заданного по варианту контекстно-зависимого языка.

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

1. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ №4

Тема работы: «Разработка контекстного анализатора».

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

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

Задание по лабораторной работе. Для предложенного варианта контекстного условия расширить атрибутную грамматику из лабораторной работы № 3 добавлением атрибутов, правил их вычисления, правил вычисления контекстных условий.

Содержание отчета:

1) титульный лист;

2) текст задания, включающий вариант задания;

3) атрибутная грамматика с атрибутами и действиями по вычислению контекстных условий;

4) исходный текст синтаксического анализатора;

5) результаты тестирования.

Методические рекомендации к лабораторной работе

Включить в программу синтаксического анализатора из лабораторной работы № 3 действия по вычислению атрибутов и проверки контекстных условий.

Варианты индивидуальных заданий

В таблице 1.1 приведены варианты заданий.

Таблица 1.1. Варианты заданий

1	Все идентификаторы должны быть разными	2	Все числа должны быть разными	3	Все идентификаторы должны быть разными
4	Все числа должны быть разными	5	Все идентификаторы должны быть разными	6	Все числа должны быть разными
7	Все числа должны быть разными	8	Все идентификаторы должны быть разными	9	Все числа должны быть разными
10	Все числа должны быть разными	11	Все числа должны быть разными	12	Все идентификаторы должны быть разными
13	Все идентификаторы должны быть разными	14	Внутри скобок все идентификаторы должны быть разными	15	Все идентификаторы должны быть разными

16	Все числа должны быть разными	17	Все идентификаторы должны быть разными	18	Все идентификаторы должны быть разными
19	Все идентификаторы должны быть разными	20	Все идентификаторы должны быть разными	21	Все числа должны быть разными
22	Все числа должны быть разными	23	Все числа должны быть разными	24	Все идентификаторы должны быть разными
25	Все числа должны быть разными	26	Все идентификаторы должны быть разными	27	Все числа должны быть разными

Контрольные вопросы:

1. Контекстно-зависимые грамматики.
2. Классификация контекстных условий языков программирования.
3. Способы формального описания контекстных условий.
4. Атрибутный метод Кнута. Атрибутная грамматика.
5. Рекомендации по формулировке контекстных условий и расстановке атрибутов.
6. Конструирование контекстных анализаторов.
7. Документирование контекстных условий и семантики.
8. W – грамматики Ван Вейнгаардена.

2. КОНТЕКСТНО-ЗАВИСИМАЯ ГРАММАТИКА

Контекстно-зависимая грамматика (КЗ-грамматика, контекстная грамматика) – частный случай формальной грамматики (тип 1 по иерархии Хомского), у которой левые и правые части всех продукций могут быть окружены терминальными и нетерминальными символами.

Другими словами, формальная грамматика G = (N, T, S, P) является контекстно-зависимой, если все правила P имеют вид:

$$\alpha A \beta \rightarrow \alpha \omega \beta$$

где $A \in N$ (то есть одиночный нетерминальный символ);

$\omega \in (N \cup T)+$ (то есть непустая цепочка, состоящая из терминальных и/или нетерминальных символов);

$\alpha, \beta \in (N \cup T)^*$ (то есть любая цепочка, состоящая из терминальных и/или нетерминальных символов).

Пример. Следующая грамматика задает контекстно-зависимый язык $\{a^n b^n c^n : n \geq 1\}$:

1. $S \rightarrow aSBC$ 4. $HB \rightarrow HC$ 7. $bB \rightarrow bb$

2. $S \rightarrow aBC$ 5. $HC \rightarrow BC$ 8. $bC \rightarrow bc$

3. $CB \rightarrow HB$ 6. $aB \rightarrow ab$ 9. $cC \rightarrow cc$

Так выглядит цепочка порождения aaabbbccc: $S \Rightarrow 1$ aSBC $\Rightarrow 1$ aaSBCBC $\Rightarrow 2$ aaaBCBCBC $\Rightarrow 3$ aaaBHBCBC $\Rightarrow 4$ aaaBHCCBC $\Rightarrow 5$ aaaBBCCBC $\Rightarrow 3$ aaaBBCHBC $\Rightarrow 4$ aaaBBCHCC $\Rightarrow 5$ aaaBBCBCC $\Rightarrow 3$ aaaBBHBCC $\Rightarrow 4$ aaaBBHCCC $\Rightarrow 5$ aaaBBBCCC $\Rightarrow 6$ aaabBBCCC $\Rightarrow 7$ aaabbBCCC $\Rightarrow 7$ aaabbbCCC $\Rightarrow 8$ aaabbbcCC $\Rightarrow 9$ aaabbbccC $\Rightarrow 9$ aaabbbccc.

Задача установления принадлежности некоторой строки s языку L(G) некоторой контекстно-зависимой грамматики G является PSPACE-полной. Это делает контекстно-зависимые грамматики абсолютно неработоспособными для практического использования.

3. АТРИБУТНАЯ ГРАММАТИКА

Атрибутная грамматика представляет собой обобщение КС-грамматики, в которой каждый грамматический символ может иметь связанное множество атрибутов, разделенное на два подмножества, – синтезируемые и наследуемые.

Атрибут может представлять собой все, что угодно, – строку, число, тип, адрес памяти и т.д. Значение атрибута в узле дерева разбора определяется семантическими правилами, связанными с используемой в данном узле продукцией.

Синтезируемые атрибуты

Синтезируемые атрибуты очень часто используются на практике. Атрибутная грамматика, использующая только синтезируемые атрибуты, называется S-атрибутной. Дерево разбора для S-атрибутной грамматики всегда может быть аннотировано путем выполнения семантических правил для атрибутов в каждом узле снизу вверх, от листьев к корню.

Рядом с именем синтезируемого атрибута пишется знак $\uparrow$.

Наследуемые атрибуты

Наследуемые атрибуты представляют собой атрибуты, значения которых в узле дерева разбора определяются атрибутами родительского и/или дочерних по отношению к родительскому узлов. Наследуемые атрибуты удобны для выражения зависимости конструкций языка от контекста, в котором они появляются. Например, мы можем использовать наследуемые атрибуты для отслеживания, появляется ли идентификатор слева или справа от знака присвоения, чтобы определить, потребуется ли адрес или значение данного идентификатора. Хотя всегда можно переписать атрибутную грамматику таким образом, чтобы использовать только синтезируемые атрибуты, зачастую более естественно воспользоваться атрибутной грамматикой с наследуемыми атрибутами.

Рядом с именем наследуемого атрибута пишется знак ↓.

Пример. Атрибутная грамматика, представленная в таблице 3.1, позволяет решить проблему распознавания последовательностей вида $a^n b^n c^n$.

Таблица 3.1. Атрибутная грамматика языка $\{a^n b^n c^n : n \geq 1\}$

Продукция	Семантические правила
$S \rightarrow ABC$	B.↓size := A.↑size; C.↓size := A.↑size
$A \rightarrow a$	A.↑size := 1
$A \rightarrow aA_1$	A.↑size := A_1.↑size + 1
$B \rightarrow b$	Условие: B.↓size = 1
$B \rightarrow bB_1$	B_1.↓size := B.↓size – 1
$C \rightarrow c$	Условие: C.↓size = 1
$C \rightarrow cC_1$	C_1.↓size := C.↓size – 1

В данном решении используется синтезируемый атрибут ↑size для последовательности символов a (нетерминал A) и наследуемый атрибут ↓size для последовательностей символов b и c (нетерминалы B и C). Мы синтезируем размер последовательности a в корне дерева разбора, присваиваем это значение атрибутам ↓size последовательностей b и c и наследуем его вниз по дереву, уменьшая значение на один всякий раз, когда видим другой символ в последовательности. Когда мы достигаем узел, в котором последовательность имеет потомка, состоящего из отдельного символа, мы проверяем, равен ли наследуемый атрибут ↓size единице. Если это так, то размер последовательности должен быть таким же, как и размер последовательности a; иначе размеры не совпадают, и разбор неуспешен.

4. ПРОГРАММИРОВАНИЕ КОНТЕКСТНЫХ АНАЛИЗАТОРОВ

Для создания контекстного анализатора берется атрибутная грамматика, в которой учтены все контекстные правила языка.

Программирование контекстных анализаторов заключается в написании своей процедуры на каждый нетерминальный символ грамматики. Атрибуты, которые соответствуют нетерминальному символу, оформляются в виде формальных параметров для соответствующей процедуры. Синтезируемые атрибуты должны передаваться по ссылке (модификатор var перед объявлением параметра в языке Pascal), а наследуемые – по значению. Все другие атрибуты, использованные в тех или иных процедурах, должны быть объявлены как переменные.

Ниже приведен исходный текст программы анализатора языка $\{a^n b^n c^n : n \geq 1\}$:

```
procedure S;
```

```
var aSize, bSize, cSize: Integer;
begin
   A(aSize);
   bSize := aSize;
   B(bSize);
   cSize := aSize;
   C(cSize);
end;

procedure A(var size: Integer);
begin
   Match('a');
   if CurWord = 'a' then begin
      // Продукция A→aA
      A(size);
      size := size + 1;
   end
   else
      // Продукция A→a
      size := 1;
end;

procedure B(size: Integer);
begin
   Match('b');
   if CurWord = 'b' then
      // Продукция B→bB
      B(size - 1)
   else
      // Продукция B→b
      if size <> 1 then
         Error;
end;

procedure C – аналогично B.
```

5. ПРИМЕР ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В данной главе приводятся результаты выполнения лабораторной работы при следующей КС-грамматике:

$S \rightarrow A :- B .$

$A \rightarrow <2> (<1>)$

$B \rightarrow A , B$

$B \rightarrow A$

и следующем контекстном условии: все числа должны быть разными.

В таблице 5.1 приведена атрибутная грамматика, обеспечивающая заданное контекстное условие.

Таблица 5.1. Атрибутная грамматика

Продукция	Семантические правила
$S \rightarrow A :- B .$	Условие: $A.\uparrow number \notin B.\uparrow numbers$
$A \rightarrow <2> (<1>)$	$A.\uparrow number := <1>$
$B \rightarrow A , B_1$	Условие: $A.\uparrow number \notin B_1.\uparrow numbers$ $B.\uparrow numbers := \{A.\uparrow number\} \cup B_1.\uparrow numbers$
$B \rightarrow A$	$B.\uparrow numbers := \{A.\uparrow number\}$

На рисунке 5.1 приведен пример работы программы.

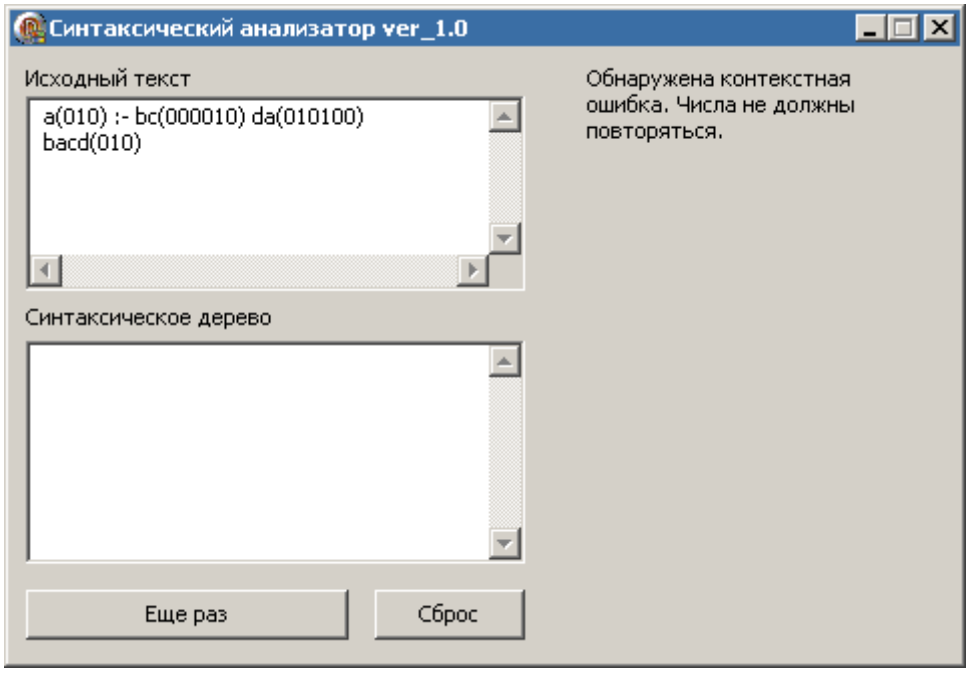

Рисунок 5.1. Пример работы программы

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.