Теория формальных языков и компиляторов
Часть 1. Порождающие грамматики и языки

Лекция 7. Эквивалентность и однозначность грамматик

7.1 Эквивалентные грамматики

При решении упражнений на прямую задачу формальных грамматик мы заметили, что можно подобрать множество грамматик, порождающих дин и тот же язык. Такие грамматики называются *эквивалентными*.

Определение 7.1. Грамматики G и G1 являются *эквивалентными*, если они порождают один и тот же язык, то есть имеет место тождество множеств:

$$L(G) = L(G1).$$

Определение 7.2. Грамматики G и G1 называются *почти эквивалентными*, если порождаемые ими языки отличаются не более чем на пустую цепочку ε, то есть имеет место:

$$L(G) = \{ L(G1) \cup \{\varepsilon\}\}.$$

Пример 7.1: Пусть грамматика $G_1[A]$ задана набором:

$V_T = \{0,1\}$,

$V_N = \{A\}$,

P: $A \rightarrow 01$

$A \rightarrow 0A1$

или через альтернативную конструкцию: $A \rightarrow 01 \mid 0A1$.

Определим язык, порождаемый грамматикой $G_1[A]$, т.е. $L(G_1[A])$. Для этого продемонстрируем процедуру выводимости в соответствии с правилами вывода P:

$$\begin{array}{ccccccc} A & \Rightarrow & 0A1 & \Rightarrow & 00A11 & \Rightarrow & 000A111 \ \ldots \\ \Downarrow & & \Downarrow & & \Downarrow & & \\ 01 & & 0011 & & 000111 \ \ldots & & \end{array}$$

Очевидно, что это множество включает вполне упорядоченный набор цепочек: { 01,0011,000111,...}. Отсюда легко видеть, что:

$$L(G1[A]) = \{0^n1^n \mid n>=1\}.$$

Рассмотрим еще одну грамматику $G_2[A]$, которая имеет свои правила P:

$$A \to 0B1$$

$$0B \to 00B1$$

$$B \to \varepsilon.$$

Тем же способом выводимости можно определить язык L(G2[A]):

$$\begin{array}{ccccccc} A & \Rightarrow & 0B1 & \Rightarrow & 00B11 & \Rightarrow & 000B111 \quad \dots \\ & & \Downarrow & & \Downarrow & & \Downarrow \\ & & 01 & & 0011 & & 000111 \quad \dots \end{array}$$

Получили тот же набор строк: { 01,0011,000111,...}, значит:

$$L(G2[A]) = \{0^n1^n \mid n>=1\}.$$

Таким образом, мы показали эквивалентность грамматик $G_1[A]$ и $G_2[A]$.

К сожалению, проблема эквивалентности грамматик в общем случае неразрешима. Это значит, что в принципе невозможно придумать алгоритм, который бы проверил эквивалентность двух грамматик. Кроме того, не все эквивалентные грамматики можно использовать для реализации языкового процессора. Реализовывать можно только *однозначные* грамматики.

7.2 Однозначные грамматики

Однозначность означает существование единственного синтаксического дерева для одной основы или одной сентенциальной формы. Проиллюстрируем проблему однозначности на конкретном примере.

Пример 7.2: Пусть грамматика подкласса арифметических выражений G[A] задана следующим альтернативным правилом:

$$A \to A+A \mid A*A \mid (A) \mid a$$

Для простоты разбора не будем учитывать символы ‘/’ и ‘-‘, а так же терминальные символы b, c. Это усечение не означает неполноту грамматики, и не нарушает её общность. По определению грамматика G[A] является эквивалентной ранее рассмотренной грамматике G[<AB>], хотя строго математически эквивалентность в этом случае не показана. Просто очевидным является порождение этими грамматиками одного и того же множества цепочек арифметических выражений.

Рассмотрим сентенциальную форму (строку): «a+a∗a». Можно построить два синтаксических дерева, показывающие два варианта разбора этой строки по одной грамматике:

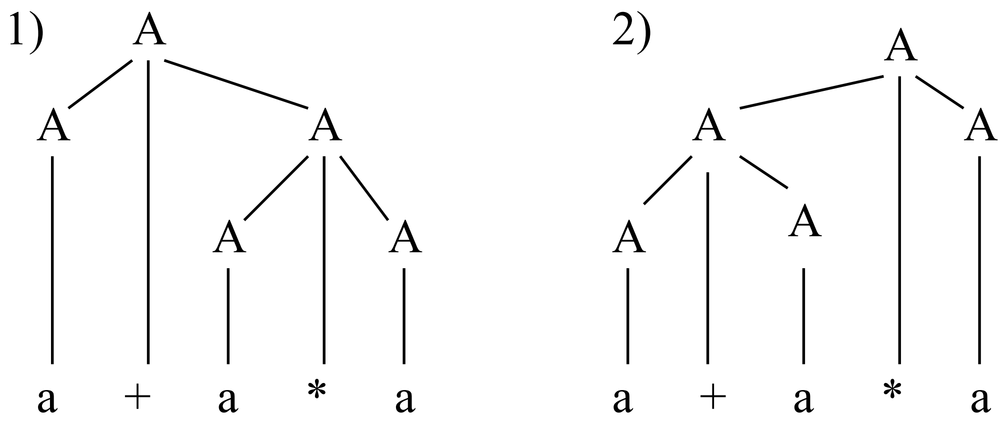

Рис. 7.1. Неоднозначность

Два синтаксических дерева для одной и той же основы (строки) и есть неоднозначность разбора. К чему может привести неоднозначность разбора? Пусть результат вычисления выражения связан с корнем синтаксического дерева. При этом в каждом узле дерева можно вычислить промежуточный результат R. Сворачивая (вычисляя) нетерминалы снизу вверх, получим:

1) a∗ a = A	2) a + a =A
a =A	A = a
R=A +A	R=A∗A

Таким образом, для разных синтаксических деревьев с одной и той же основой получены разные результаты R. Неоднозначная грамматика не предполагает семантического единства и верность синтаксического разбора.

Проблема однозначности и безвозвратности разбора является важной задачей системного программирования в части проектирования языковых процессоров и состоит в следующем. Безвозвратность предполагает спуск по синтаксическому дереву без подъёмов или возвратов. Если подобранная грамматика G удовлетворяет требованиям языка ею порождаемого L(G), но при этом не является однозначной, то необходимо, если это возможно, подобрать эквивалентную однозначную грамматику G1.

7.3 Критерии неоднозначности

Выше уже отмечалось, что проблемы эквивалентности и однозначности алгоритмически в общем случае неразрешимы, лишь в частом случае можно добиться успеха. Удалось, однако, выделить множество продукций, по наличию которых в правилах вывода грамматик P можно судить об однозначности. Такие правила называются *критериями неоднозначности*. Критерии неоднозначности показывают такие формы записи правил грамматики, которые приводят к неоднозначности грамматики.

Итак, грамматика неоднозначна, если она содержит хотя бы одно правило следующего вида:

1) $H \to HH \mid \alpha$

2) $H \to H\,\alpha\,H \mid \beta$

3) $H \to \alpha\,H \mid H\,\beta \mid \gamma$

4) $H \to \alpha\,H\,\beta\,H$,

где $\alpha, \beta, \gamma \in V^+$, $V = (V_T \cup V_N)^+$.

В рассмотренном выше примере G[A] есть альтернативные продукции:

$$A*A \mid A/ A \mid A\text{-}A \mid A+A \mid a$$

Эти правила по форме записи похожи на второй критерий неоднозначности. Именно поэтому удалось легко показать неоднозначность G[A].

Упражнения

1. Определите, является ли однозначной грамматика:

$$G[I]: P = \{I \to AA,\ A \to a,\ A \to aa\}.$$

Если грамматика неоднозначная, найдите несколько строк, для которых можно построить два и более синтаксических деревьев. Приведите примеры неоднозначности разбора этих строк.

2. Определите, является ли однозначной грамматика:

$$G[I]: P = \{I \to aABc,\ I \to \$,\ A \to cIB,\ A \to Ab,\ B \to bB,\ B \to a\}.$$

Если грамматика неоднозначная, найдите несколько строк, для которых можно построить два и более синтаксических деревьев. Приведите примеры неоднозначности разбора этих строк.

3. Пусть A = {0, 1}. Построить грамматику G, такую что^

$$L(G) = \{\alpha\alpha^R,\ \alpha \in A^*\}.$$

4. Для заданной грамматики построить эквивалентную грамматику без цепных правил (вида $A \to B$):

$$G[I]: P = \{I \to aM,\ M \to A,\ A \to aA \mid B,\ B \to bB \mid b\}.$$

Список использованных источников

1. Шорников Ю.В. Теория и практика языковых процессоров.