Лекция 13

Неоднозначные грамматики. Способы устранения неоднозначности

- Напомним, что КС-грамматика G является неоднозначной, если существует цепочка w ∈ L(G), имеющая два или более различных левых или правых вывода. Если грамматика используется для определения языка программирования, желательно, чтобы она была однозначной. В противном случае программист и компилятор языка могут по-разному понять смысл некоторых программ.

- *Пример 13.1. Самый известный пример неоднозначности в языках программирования - это "кочующее else". Рассмотрим грамматику с правилами*
- *S → if b then S else S*
- *S → if b then S*
- *S → a*
- *Эта грамматика неоднозначна, так как в ней цепочка*
- *if b then if b then a else a имеет два левых вывода:*
- *S ⇒ if b then S else S ⇒ if b then if b then S else S⇒ if b then if b then a else S*
- *⇒ if b then if b then a else a*

- *u S ⇒ if b then S ⇒ if b then if b then S else S⇒ if b then if b then a else S*
- *⇒ if b then if b then a else a.*
- *Эти выводы предполагают разную интерпретацию if b then (if b then a else a) или if b then (if b then a) else a.*

- Определенная неоднозначность - это свойство грамматики, а не языка. Для некоторых неоднозначных грамматик можно построить эквивалентные им однозначные грамматики.

- *Пример 13.2. Построим эквивалентную грамматику для грамматики примера 13.1.*
- *S1 → if b then S1*
- *S1 → if b then S2 else S1*
- *S1 → a*
- *S2 → if b then S2 else S2*
- *S2 → a*
- *Здесь очевидно грамматика является однозначной. Неоднозначность исходной грамматики можно устранить, договорившись, что else должно ассоциироваться с последним из предшествующих ему then (как это принято в языках программирования).*

- Общего алгоритма, выясняющего, однозначна ли грамматика, не существует. Приведем несколько наиболее распространенных конструкций, приводящих к неоднозначности, и способов ее устранения.
- 1. Пусть грамматика содержит правила
- $A \rightarrow AA$
- $A \rightarrow a$
- Очевидно, что такая грамматика будет неоднозначной. Эта неоднозначность устраняется следующей грамматикой
- $A \rightarrow AD$
- $A \rightarrow D$
- $D \rightarrow a$

- или
- $A \rightarrow DA$
- $A \rightarrow D$
- $D \rightarrow a$

- 2. Следующий пример правил, приводящих к неоднозначности
- $A \rightarrow AaA$
- 3. Грамматика, содержащая следующие правила, также неоднозначна
- $A \rightarrow aA$
- $A \rightarrow Ab$

- 4. Еще один пример неоднозначной грамматики
- $A \rightarrow aAbA$
- $A \rightarrow aA$
- Во всех рассмотренных случаях введение нового нетерминального символа позволяет устранить неоднозначность.

Определение 13.1

- КС-язык называется существенно неоднозначным, если он не порождается никакой однозначной КС-грамматикой.
- В примере 8.1.приведена неоднозначная грамматика, порождающая арифметическое выражение
- $E \rightarrow E+E \mid E*E \mid (E) \mid a$
- Эту неоднозначность можно устранить, взяв вместо G грамматику G1 со схемой:
- $E \rightarrow E+T \mid E*T \mid T$
- $T \rightarrow (E) \mid a$
- либо грамматику G2
- $E \rightarrow E+T$
- $T \rightarrow T*F \mid F$
- $F \rightarrow (E) \mid a$

Теорема 13.1

- Проблема, однозначна ли КС-грамматика, неразрешима.
- Рассмотрим еще один пример грамматики, порождающей оператор присваивания (приведем только те правила, которые приводят к неоднозначности).
- <оператор присваивания> → <левая часть> := <выражение>
- <левая часть> → <идентификатор функции>
- <левая часть> → <идентификатор переменной>
- <идентификатор функции> → <идентификатор>
- <идентификатор переменной> → <идентификатор>
- <идентификатор> → A<символ идентификатора>
- <символ идентификатора> → 1

- <символ идентификатора> → 2
- <выражение> → <арифметическое выражение>
- <выражение> → <символьное выражение>
- <арифметическое выражение> → <указатель функции>
- <символьное выражение> → <указатель функции>
- <указатель функции> → <идентификатор функции>(<аргументы функции>)
- Легко видеть, что данная грамматика является неоднозначной. В данном случае можно построить однозначную грамматику, порождающую тот же язык.

- <оператор присваивания> → <левая часть> := <выражение>
- <левая часть> → <идентификатор>
- <идентификатор> → A<символ идентификатора>
- <символ идентификатора> → 1
- <символ идентификатора> → 2
- <выражение> → <указатель функции>
- <указатель функции> → <идентификатор>(<аргументы функции>)

- Первая грамматика содержит нетерминальные символы, определяющие "смысл" идентификатора, во второй грамматике "смысл" отсутствует. Тем не менее, при разборе цепочек языка необходимо знать, какой смысл имеет используемый идентификатор. Эта проблема решается определением контекстных условий для языка программирования. Контекстные условия языков программирования формулируются с помощью естественного языка. Приведем классификацию контекстных условий языка (см. [3])

- (1) Контекстные условия о правилах описания идентификаторов в программах. Сюда относятся контекстные условия следующих типов: все используемые в программах идентификаторы должны быть описаны до их использования в программе; каждый из идентификаторов, используемых в программе, должен быть описан один раз (для каждого идентификатора обычно определяется его область действия, и единственное описание должно относиться к этой области действия - блоку программы, модулю программы, подпрограммы, описание в формальных параметрах процедур и функций и т.д.).

- (2) Контекстные условия о правилах использования идентификаторов в своей области действия, т.е. контекстные условия, задающие соответствие определяющего и использующего вхождений идентификаторов. Здесь под определяющим вхождением понимается вхождение идентификатора в конструкцию, описывающую этот идентификатор, а использующим - вхождение идентификатора в конструкцию, которая не рассматривается как его описание в языке (например, вхождение идентификатора в выражение).

- (3) Контекстные условия, определяющие правила соответствия видов величин, входящих в синтаксические конструкции программ. Сюда относятся контекстные условия о соответствии формальных и фактических параметров процедур и функций, о соответствии величин, используемых в выражениях (например, говорящие о том, что нельзя складывать число и строку, нельзя, чтобы результатом условия было числовое значение и т.д.), о соответствии количества формальных и фактических параметров процедур и функций, о соответствии размерности массива при описании и при использовании и другие.

- (4) Контекстные условия, задающие различные количественные ограничения, их обычно называют ограничениями реализации. Сюда можно отнести контекстные условия о вложенности скобок в выражении, о допустимом количестве идентификаторов в программах и т.д.

- В конкретном языке программирования необязательно присутствие контекстных условий каждой группы. Какие контекстные условия задать для языка определяет разработчик языка или разработчик компилятора этого языка (последнее обычно относится к контекстным условиям четвертой группы). Средством формального описания контекстных условий языков программирования являются атрибутные грамматики, которые часто используются при разработке компилятора языка программирования в системах построения трансляторов (определение атрибутных грамматик см. в [5]). Другим примером грамматик, которые могут использоваться при описании контекстных условий языков программирования, являются грамматики ван Вейнгаардена (см. [3,5]).