Построение грамматик и грамматики, описывающие основные конструкции языков программирования

Задача построения формальных грамматик состоит в том, что для множества конечных цепочек, заданного в виде описания на естественном языке, требуется построить формальную грамматику, порождающую это множество цепочек. Учитывая, что терминальный словарь грамматики должен включать все символы, используемые для построения цепочек, входящих в заданное множество, результатом решения задачи должны явиться нетерминальный словарь и схема грамматики.

Построение этих объектов представляется достаточно сложным, поскольку оно должно выполняться неформально и требует мысленного охвата всевозможных допустимых вариантов цепочек заданного множества и синтеза правил их построения. Построение усложняется еще и тем, что оно, как и всякая другая задача синтеза, имеет много решений.

Рекомендации по построению грамматик

Основой создания правил грамматики является способ выделения структуры заданного множества цепочек. Этот способ предусматривает расчленение цепочек, входящих в заданное множество, на части таким образом, чтобы выявить повторяющиеся части цепочек и части, входящие во все цепочки в неизменном виде. Такое расчленение на части представляет собой выявление структуры цепочек заданного множества.

Для каждого выявленного элемента структуры введем обозначения. Множество таких обозначений составляет основу словаря нетерминальных символов некоторой грамматики.

Следующим шагом построения является выявление последовательностей, в которых элементы структуры могут входить в заданные цепочки. Такие последовательности являются основой для построения правил грамматики.

Чтобы показать, каким образом структура цепочек отображается в правила грамматики, рассмотрим следующие примеры.

1. Цепочке, состоящей из заданных символов **abc**, соответствует правило

A → abc

2. Цепочке, начинающейся с заданного символа **a**, соответствует правило

A → aB

3. Цепочке, заканчивающейся заданным символом **a**, соответствует правило

A → Ba

4. Цепочке, начинающейся и заканчивающейся заданными символами **a**, **b**, соответствует правило

A → aBb

5. Цепочке, содержащей в середине символ **a**, соответствует правило

A → BaC

6. Цепочке заданной длины **l =2** соответствуют правила:

A → aB и B → a

7. Цепочке, состоящей из повторяющихся символов **a**, соответствуют правила

A → aA и A → a

8. Цепочке, состоящей из чередующихся символов **a** и **b**, соответствуют правила:

A → aB и B → bA

Описание списков

В качестве первых примеров рассмотрим построение грамматик для последовательностей символов и последовательностей символов с разделителями. Такие последовательности часто называют **списками**.

1) Обозначим элемент последовательности **a**. Простейшая последовательность может состоять из одного элемента **a**. Все другие последовательности могут быть получены путем приписывания к уже построенной последовательности еще одного элемента. Если обозначить построенную часть последовательности нетерминальным символом **R**, а последовательность символом **L**, то получим правила грамматики в виде:

L → aR

R → aR

R → ε

2) В предыдущей задаче предполагалось, что список **L** должен содержать хотя бы один элемент. Если же допустить, что множество цепочек, порожденных правилами грамматики, может включать пустой символ, то к построенным правилам нужно добавить еще одно правило **L → ε**. В этом случае набор правил имеет вид:

L → aR

R → aR

R → ε

L → ε

1. Рассмотрим построение списка, между элементами которого должны стоять разделители. Выберем в качестве разделителя запятую. Простейший список, как и в предыдущем случае, состоит из одного элемента, а построение списка из нескольких элементов может быть выполнено приписыванием к уже построенной части списка разделителя с элементом списка. Правила, соответствующие этому построению, имеют вид:

L → aR

R → ,aR

R → ε

4) Если список с разделителями может быть пустым, то приведенный выше набор правил нужно дополнить еще одним правилом с пустой правой частью. В результате получим:

L → aR

R → ,aR

R → ε

L → ε

В общем случае, если описано множество цепочек, представляющих собой некоторый язык, и требуется построить грамматику, порождающую это множество цепочек, то следует поступать так:

1. Выписать несколько примеров из заданного множества цепочек.
2. Проанализировать структуру цепочек, выделяя начало, конец, повторяющиеся символы или группы символов.
3. Ввести обозначения для сложных структур, состоящих из групп символов. Такие обозначения являются нетерминальными символами искомой грамматики.
4. Построить правила для каждой из выделенных структур, используя для задания повторяющихся структур рекурсивные правила.
5. Объединить все правила.
6. Проверить с помощью выводов возможность получения цепочек с разной структурой.

ЗАДАНИЕ

1. *Грамматики для целых чисел без знака и идентификаторов*

*Идентификатор*

*Тройной идентификатор*

1. *Грамматики для арифметических выражений*

*Использование скобок без вложенности*

*Вложенные скобки*

1. *Грамматика для последовательности операторов присваивания*
2. *Грамматики для условных операторов и операторов цикла*

**ДЗ**

1. **Грамматика для описания правильных скобочных выражений**
2. **Грамматика для описания правильных автомобильных номеров**
3. **Грамматика для описания квадратных уравнений**

**Грамматики для целых чисел без знака и идентификаторов**

Целые числа представляют собой последовательность цифр, поэтому их можно рассматривать как списки, элементами которых являются цифры. Используя в качестве аналога грамматику, задающую список без разделителей, получаем схему грамматики для целых чисел в виде:

N → DR

R → DR

R → ε

D → 0 | 1 | ... | 9

Структуру идентификатора можно представить в виде двух компонентов: начала и основной части. Началом может быть любая из букв, а основная часть представляет собой список без разделителей, элементами которого могут быть либо буквы, либо цифры. Используя выделенные компоненты, получаем схему грамматики вида:

I → CA

A → CA | DA

A → C | D

A → ε

D → 0 | 1 | ... | 9

C → a | d | c | ... | z

Если наложить ограничения на длину идентификатора, например, допустить использование идентификаторов, состоящих только из трех символов, то схема грамматики получается проще.

I → C A1

A1 → CB | DB

B → C | D

D → 0 | 1 | ... | 9

C → a | d | c | ... | z

Грамматики для арифметических выражений

Условимся рассматривать арифметические выражения, использующие только знаки сложения и умножения. Вначале построим грамматику для выражений без скобок. Такие выражения представляют собой цепочки, которые можно рассматривать как списки с разделителями, в которых роль разделителей выполняют знаки операций. В соответствии с этой аналогией получаем схему грамматики, в которой символ **I** обозначает идентификатор, определение которого приведено выше.

H → IR

R → W I R | ε

W → + | \*

Чтобы построить грамматику, допускающую использование в арифметических выражениях скобок без вложенности, представим структуру таких выражений в виде списка с разделителями, элементами которого являются выражения без скобок или выражения без скобок, заключенные в скобки. Разделителями этого списка являются знаки операций. Такой структуре соответствует следующая схема грамматики:

G → HQ

G → (H)Q

Q → WG | ε

Для построения грамматики арифметических выражений, допускающих применение вложенных скобок, следует предусмотреть возможность использования в качестве элемента списка не только выражения без скобок, но и выражения, в котором могут быть использованы скобки. Схема грамматики, учитывающая эту возможность, может быть записана в виде:

E → GP | (E)P

P → WE | ε

Грамматика для описаний

Пусть требуется построить грамматику для описания целых и вещественных переменных. Описание переменных определенного типа должно начинаться указателем типа **'real'** или **'int'**. В полном тексте описания переменных определенного типа могут повторяться. Например, полное описание может включать три разных описания переменных целого типа. Полное описание должно заканчиваться точкой. В качестве разделителя описаний переменных разных типов примем точку с запятой, а в качестве разделителя переменных одного типа - запятую. Структуру полного описания можно представить в виде двух вложенных списков с разделителями. Внутренний список, рассматриваемый как элемент внешнего списка, представляет собой описание переменных одного типа. Он имеет заголовок в виде указателя типа, за которым следует последовательность идентификаторов, разделенных запятыми. Внешний список использует в качестве разделителя точку с запятой. Схема грамматики рассматриваемого вида может быть записана так:

Z → A2

A2 → B1 C1

C1 → ; B1 C1 | ε

B1 → 'real' L | 'int' L

L → I K

K → , I K | ε

Грамматика для последовательности операторов присваивания

Допустим, что в правой части оператора присваивания могут быть использованы только выражения без скобок (см. выше). В качестве разделителя операторов в последовательности примем точку с запятой. Учитывая, что последовательность операторов соответствует списку с разделителями в виде точки с запятой, и, используя определенную ранее конструкцию идентификатора, получаем искомую схему грамматики в виде:

U → A3

A3 → S R1

R1 → ; S R1 | ε

S → I B2

B2 → := H1

H1 → I R2

R2 → + I R2 | \* I R2 | ε

Грамматики для условных операторов и операторов цикла

Допустим, что рассматриваются условные операторы, аналогичные используемым в языке Паскаль, с разделителями 'if', 'then', 'else'. В качестве условия в таких операторах разрешается использовать отношения, состоящие из двух идентификаторов, соединенных знаками = и <. Структура такого оператора определяется двумя видами последовательностей фиксированной длины, для описания которых можно воспользоваться простым перечислением компонентов. Первая последовательность определяет полный и сокращенный условные операторы, а вторая - конструкцию "отношение". Схема грамматики, задающая эти последовательности, может быть изображена так:

V> → .if. R4 C2

C2 → .then. S C3

C3 → .else. S | ε

R4 → I R3

R3 → < I | = I

В этой грамматике **S** определяется схемой предыдущей грамматики. Рассмотрим описание операторов цикла, подобных используемым в языке Паскаль, с разделителями **'while'**, **'do'**, **'repeat'**, **'until'**. Каждый оператор может быть описан в виде простой последовательности ограниченной длины, в которой используются построенные ранее грамматики для определения понятий R4 и S. На практике часто встречаются ситуации, когда удобнее работать с грамматикой, правая часть правил которой начинается терминальным символом. Построение подобных грамматик сводится к тому, что для каждого терминального символа заданной конструкции языка строится отдельное правило. Для рассматриваемых операторов цикла такие схемы грамматик с использованием определенных ранее нетерминальных символов имеют вид:

W → .while. R4 C4

C4 → .do. S

W1 → .repeat. S C5

C5 → .until. R4