Теория формальных языков и компиляторов

Часть 1. Порождающие грамматики и языки

Лекция 3. Порождающие грамматики

Первым способом определения языка, который изучается в нашем курсе, являются порождающие грамматики. Что такое грамматика?

В широком смысле, грамматика – это раздел науки «лингвистика», в которой изучается строение слов (словообразование) и способы изменения слов (морфология), виды словосочетаний и типы предложений (синтаксис).

В теории формальных языков грамматика – это способ описания формального языка. Поскольку язык – это *множество* слов, то грамматика – это способ задания (определения элементов). Различают порождающие и распознающие (аналитические) грамматики. Первые задают правила, с помощью которых можно построить любое слово (или строку) языка, а вторые позволяют по заданному слову определить, входит ли оно в язык или нет. Здесь изучаются порождающие грамматики. Поэтому *грамматикой* будем называть набор правил, определяющих способ построения текстов на некотором языке.

3.1 Запись правил грамматики

Грамматика имеет свой метаязык, т.е. язык для описания другого языка. Впервые такой метаязык был предложен Н. Хомским для описания естественных языков.

Один из наиболее распространенных способов описания синтаксиса языка – это форма Бэкуса-Наура (Backus-Naur-Form, БНФ). Этот способ был разработан Дж. Бэкусом и П. Науром для описания грамматики языка Алгол-60. В дальнейшем способ Бэкуса-Наура был использован для многих других языков.

При записи грамматики в форме Бэкуса-Наура используются два типа объектов:

- основные символы (терминальные символы и, в частности, ключевые слова);
- металингвистические переменные (нетерминальные символы), значениями которых являются цепочки основных символов описываемого языка.

Металингвистические переменные изображаются словами (русскими или английскими), заключенными в угловые скобки (< >) и разделённые металингвистическими связками (::= и |). Знак ::= означает выводимость, следование. То есть из того, что стоит слева от знака следует то, что стоит справа. Проще говоря, левая часть правила заменяется на правую. Знак | разделяет альтернативные, взаимозаменяемые варианты.

Пример 3.1: Пусть в некотором языке возможна только такая структура предложений, где первым всегда идет прилагательное, за ним – подлежащее, потом – сказуемое, и в конце – существительное. Тогда грамматику языка можно задать таким правилом:

```
<предложение> ::= <прилагательное> <подлежащее>
                  <сказуемое> <существительное>
```

При определении синтаксиса языков Pascal и Modula-2 Н. Вирт использовал расширенную форму Бэкуса-Наура (EBNF), в которой:

- Нетерминальные символы записываются как отдельные слова.
- Терминальные символы записываются в кавычках, например, "BEGIN".
- Вертикальная черта (|), как и прежде, используется для определения альтернативных вариантов.
- Круглые скобки используются для группировки символов.
- Квадратные скобки используются для определения возможного вхождения символа или группы символов.
- Фигурные скобки используются для определения возможного повторения символа или группы символов (вспомните действие «итерация» над строками).

- Символ равенства используется вместо символа ::=.
- Символ точка используется для обозначения конца правила.
- Комментарии к правилам грамматики записываются между символами (* … *).

Пример 3.2: Правила для записи целых чисел:

```
Integer = Sign UnsignedInteger.
Sign = ["+"|"-"].
UnsignedInteger = digit {digit}.
digit = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".
```

Эти правила читаются так: Первое правило описывает грамматику для записи целого числа (оно обозначено нетерминальным символом Integer). Нетерминальный символ Integer нужно заменить последовательностью нетерминалов: сначала Sign, после него – UnsignedInteger. Очевидно, что Sign обозначает знак числа, а UnsignedInteger – модуль числа. Знак числа указывать необязательно, поэтому во втором правиле текст после знака = записан в квадратных скобках. Если же указывать знак, то это может быть плюс или минус. Поэтому во втором правиле символы + и – разделены вертикальной чертой. Сами символы записаны в кавычках, потому что это терминальные символы. В записи модуля числа UnsignedInteger (третье правило) обязательно должна быть хотя бы одна цифра (digit). Чтобы записывать числа из нескольких цифр в правиле есть повторяющийся фрагмент {digit}. Запись в фигурных скобках означает, что здесь может быть сколько угодно цифр. Или вообще ни одной цифры. И наконец, цифра – это любой из символов от 0 до 9, как показывает последнее правило.

В 1981 году Британский институт стандартов (British Standards Institute) опубликовал *стандарт* EBNF. В него добавили три требования:

- Элементы правил разделяются запятыми.
- Правила заканчиваются точкой с запятой.
- Пробелы не являются значащими.

Этот стандарт получился более наглядным, чем расширенная форма Бэкуса-Наура, предложенная Виртом.

Пример 3.3: Грамматика для записи числовых констант:

```
Constant Declaration = "CONST", Constant Identifier,
                       "=", Constant Expression, ";",
                       {Constant Identifier, "=",
                       Constant Expression, ";"};
```

Числовая константа – это переменная в программе, значение которой присваивается в момент ее создания. В дальнейшем нельзя изменять значение этой переменной. Ключевое слово CONST является признаком, что это переменная с неизменяемым значением.

По представленной грамматике для определения числовой константы необходимо сначала записать ключевое слово CONST, затем – имя переменной (Constant Identifier). После имени переменной должен быть знак =. После знака = записывается выражение (формула) Constant Expression для вычисления значения переменной. Заканчивается объявление константы точкой с запятой. Если нужно определить еще несколько констант, то последовательность записи будет такая же, но повторять слово CONST не требуется. Пример строки, записанной по грамматике:

```
CONST abc = 2; C1 = abc + 3; C2 = (5+8)/2;
```

3.2 Определение грамматики

Теперь обратимся к строгому определению грамматики.

Определение 3.1. Грамматикой G[Z] называется четверка объектов: 1) множество терминальных символов (терминальный алфавит или словарь); 2) множество нетерминальных символов; 3) конечное непустое множество правил вывода и 4) начального нетерминального символа Z, который должен встречаться хотя бы один раз в левой части правила вывода.

Математически определение грамматики записывается так:

$G[Z] = \{V_T, V_N, P, Z\}$, где:

- V_T – конечное множество терминальных символов;
- V_N – не пересекающееся с V_T конечное множество нетерминальных символов ($V_T \cap V_N = \varnothing$);
- P – конечный набор порождающих правил вида ($\alpha \to \beta$), где $\alpha \in V^+$, $\beta \in V^*$;
- Z – начальный символ, $Z \in V_N$;
- $V = V_T \cup V_N$ – объединение словарей (общий словарь терминалов и нетерминалов).

В дальнейшем мы будем использовать расширенную форма Бэкуса-Наура, где символ «$\to$» используется вместо символа «=».

Пример 3.4. Определить грамматику целых чисел G1[Z].

P: 1) <целое без знака> $\to$ < цифра >

2) <целое без знака> $\to$ <цифра > <целое без знака>

Следуя введённому формальному определению грамматики, представим G1[Z] её составляющими:

Z = <целое без знака>;

V_T ={0, 1, 2,..., 9};

V_N ={<целое без знака>, <цифра >}.

Пример 3.5. Пусть грамматика Хомского G2[Z] задана следующим образом:

$G2[Z] = \{V_T, V_N, P, Z\}$

V_T = {a, bc};

V_N = {A, B, C};

P = { 1) Aab $\to$ cb

2) B $\to$ ca

3) BA $\to$ c }.

Правило вывода множества P часто называют продукциями. Впервые продукцией воспользовались в своих работах Бэкус и Наур.

3.3 Выводимость

Здесь мы рассмотрим выводимость – приём, который мы будем использовать в дальнейшем для получения строк языка по заданной грамматике. Сформулируем несколько определений.

Определение 3.2. Говорят, что из строки (цепочки) α *непосредственно следует* цепочка β (обозначается $\alpha \Rightarrow \beta$), если при $\alpha = \mu\tau\nu$ и $\beta = \mu\gamma\nu$ в правилах вывода P грамматики G[Z] существует продукция, такая что $\tau \to \gamma$ (из τ следует γ).

Пример 3.6. Пусть $\alpha = BAabc$, $\beta = Bcbc$. Грамматика G3[Z] содержит правило: $Aab \to cb$. Доказать, что из α непосредственно следует β ($\alpha \Rightarrow \beta$).

Сравнивая строки α и β и пользуясь определением 3.2, можно увидеть: $\mu = B$, $\tau = Aab$, $\gamma = cb$, $\nu = c$. В G3[Z] есть правило: $Aab \to cb$ или $\tau \to \gamma$. Значит, что из α непосредственно следует β ($\alpha \Rightarrow \beta$).

Определение 3.3. Говорят, что из α *выводится* β ($\alpha \Rightarrow^* \beta$), если имеет место следующая непосредственная выводимость: $\alpha = \omega_0 \Rightarrow \omega_1 \Rightarrow \omega_2 \Rightarrow ... \Rightarrow \omega_n = \beta$, то есть n-кратная непосредственная выводимость порождает *итерационную выводимость* или просто *выводимость*.

Можно дать более компактное определение понятия выводимости и итерационной выводимости:

Определение 3.4. Если $\alpha\beta\gamma \in L(G)$ – цепочка языка, а $\beta \to \delta \in P$ – правило грамматики G, то $\alpha\beta\gamma \Rightarrow_G \alpha\delta\gamma$ ($\alpha\delta\gamma$ непосредственно выводима из $\alpha\beta\gamma$ в G).

Рефлексивное и транзитивное замыкание этого отношения обозначим как $\alpha \Rightarrow_G^* \beta$ (цепочка β итерационно выводима из α по правилам грамматики G).

Пример 3.7. Рассмотрим грамматику G4[Z = < целое без знака >] с правилами:

1) <целое без знака> $\rightarrow$ <цифра>

2) <целое без знака> $\rightarrow$ <цифра> <целое без знака>

3) <цифра> $\rightarrow 0|1|2|3|\ldots|9$

Теперь поставим задачу анализа строк: принадлежит ли цепочка «123» множеству <целых без знака >?

Анализ проведём, используя определение выводимости:

<целое без знака> $\Rightarrow^{(2)}$ <цифра> <целое без знака> $\Rightarrow^{(2)}$

$\Rightarrow^{(2)}$ <цифра> <цифра> <целое без знака> $\Rightarrow^{(1)}$

$\Rightarrow^{(1)}$ <цифра> <цифра> <цифра> $\Rightarrow^{(3)}$

$\Rightarrow^{(3)}$ «123»

Цифры в круглых скобках над стрелками выводимости соответствуют нумерации правил вывода. Итак, используя правила грамматики G4 и понятие выводимости, нам удалось убедиться в принадлежности строки 123 множеству <целых без знака>. Легко проследить с помощью той же процедуры выводимости принадлежность цепочки 0123 грамматике целых чисел. В некоторых языках программирования константы, которые начинаются с 0, относятся к недесятичной системе исчисления. Это означает, что грамматика, которая определяет синтаксис, включает также в определённых случаях и семантическую нагрузку. Для того чтобы и семантика была правильной в данном примере, преобразуем грамматику G4 в G4’[<ЦБЗ> = <целое без знака>] так, что:

G4’[<ЦБЗ>] содержит правила:

1) <ЦБЗ> $\rightarrow$ <Ц0> { <Ц0> | <цифра> } | 0

2) <Ц0> $\rightarrow$ 1|2|3|4|5|6|7|8|9

Преобразованная грамматика G4’[<ЦБЗ>] не является *эквивалентной* G4, т.к. строки, выводимые из <ЦБЗ>, не могут начинаться с 0 в естественной форме.

Определение 3.5. Грамматика является леворекурсивной, если в ней есть нетерминальный символ А такой, что существует порождение $A \Rightarrow^{*} A\alpha$.

Не все методы разбора текста могут работать с леворекурсивными грамматиками, так как практическая реализация леворекурсивных грамматик часто приводит к зацикливанию алгоритма разбора. Поэтому от левой рекурсии избавляются преобразованием исходной грамматики в *эквивалентную*, но без леворекурсивных правил. Идея такого преобразования состоит в следующем.

Пусть имеем леворекурсивную продукцию $A \to A\alpha \mid \beta$. Введём дополнительный нетерминал B такой, что:

$A \to \beta B$

$B \to \alpha B \mid \varepsilon$

Легко показать эквивалентность правил (продукций), т.е. выводимость одних и тех же цепочек. В последних двух правилах вывода отсутствует левосторонняя рекурсия. Платой при этом явилось добавление нового правила вывода и нетерминального символа. При этом гарантируется отсутствие зацикливания при разборе строк.

Упражнения

1. Можно ли записать число «0000» по грамматике из примера 3.2? Попробуйте улучшить грамматику.

2. Можно ли записывать числа в двоичной, восьмеричной или шестнадцатеричной системе счисления по грамматике из примера 3.2? Что бы вы изменили в грамматике для этого?

3. Можно ли записывать числа с десятичной дробной частью (например, 1.25 или 0.03) по грамматике из примера 3.2? Что бы вы изменили в грамматике для этого?

4. Дополните грамматику из примера 3.3 правилом для записи имени числовой константы Constant Identifier (идентификатора). Идентификатор

записывается с использованием букв и цифр, причем обязательно должен начинаться с буквы.

5. Дополните грамматику из примера 3.3 правилом для записи инициализирующего выражения Constant Expression (выражения). В качестве выражении могут использоваться имена переменных (идентификаторы), числа или формулы. В записи формул можно использовать идентификаторы и числа, а также знаки операций + и -.

6. Покажите выводимость строк «050», «12345» и «000» по грамматике G4 из примера 3.7.

7. Покажите выводимость строк «0», «123» и «98765» по грамматике G4’ из примера 3.7.

8. Выведите как можно больше (7 – 9) строк по правилам грамматики. Выведенные строки должны содержать только терминальные символы.

8.1. P: 1) $Z \to 11XY0$ 2) $X \to 1X \mid 1 \mid \varepsilon$ 3) $Y \to 1Y0 \mid \varepsilon$

8.2. P: 1) $Z \to XY - X$ 2) $X \to 1X \mid 1 \mid \varepsilon$ 3) $Y \to 1Y0 \mid \varepsilon$

8.3. P: 1) $Z \to AB$ 2) $A \to 1A0 \mid 10$ 3) $B \to 1B0 \mid 1$

8.4. P: 1) $Z \to A + B$ 2) $A \to aA \mid \varepsilon$ 3) $B \to bB \mid \varepsilon$

8.5. P: 1) $Z \to A - B$ 2) $A \to 1A00 \mid 10$ 3) $B \to aB \mid \varepsilon$

9. Запишите полные определения грамматик (как в примере 3.5), содержащих правила из упражнения 8.

Список использованных источников

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