Теория формальных языков и компиляторов

Часть 1. Порождающие грамматики и языки

Лекция 2. Обозначения и определения

2.1 Символы

Для того чтобы записать текст на некотором языке, необходим алфавит. Алфавит – это набор (множество) букв или *символов*. Вы знаете, что существуют, например, русский алфавит (а, б, в, г, …, я), английский алфавит (a, b, c , d, …, z), греческий алфавит (α, β, γ, δ, …, ζ) и т. д. Для записи чисел есть свой алфавит – цифры (0, 1, 2, 3, …, 9). Для записи чисел в двоичном формате используются только символы 0 и 1, то есть алфавит двоичной системы счисления содержит всего 2 символа. В шестнадцатеричной системе счисления числа записываются с использованием 16 символов (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

В теории формальных языков также имеется понятие «символ». Причем различают 2 вида символов: *терминальные* и *нетерминальные*.

Определение 2.1. Терминальные символы – это символы (буквы), используемые для записи текстов на формальном языке.

Название «терминальный символ» связано с понятием «терминал». Терминал – это устройство для оперативного ввода и вывода информации, используемое при взаимодействии пользователя с вычислительной машиной или сетью. Терминал – это устаревшее название монитора, который отображает только текстовую информацию, и клавиатуры. Раньше пользователь мог взаимодействовать с компьютером только одним способом – вводя команды с клавиатуры (как в «Командной строке» Windows) и читая текстовые сообщения на экране. Поэтому символы, используемые для записи команд и вывода сообщений, называются терминальными.

Как известно, языки программирования являются формальными языками. Поэтому тексты программ на языках программирования можно

рассматривать как строго упорядоченную последовательность *терминальных символов*.

Примеры терминальных символов:

- строчные буквы английского алфавита (a, b, c , d, …, z);
- прописные буквы английского алфавита (‘A’, ‘B’, ‘C’ , ‘D’, …,’Z’);
- цифры (0, 1, 2, 3, …, 9);
- знаки математических операций (+, -, *, /, %, =, <, >);
- скобки (‘(‘, ‘)’, [,], {, });
- знаки пунктуации (‘.’, ‘,’, ‘:’, ‘;’, ‘!’);
- специальные символы (~, @, #, $, &, |, \, ^);
- пробел (‘ ‘) и табуляция (‘ ‘);
- перевод строки (Enter, ¶).

Обратите внимание, что терминальные символы можно записывать в апострофах (‘’). Это делается, чтобы легко отличать терминальные символы от других символов. Рекомендуется всегда использовать апострофы в записи.

Также обратите внимание, что мы не использовали символы русского или греческого алфавита. Дело в том, что традиционно для кодировки символов применялась кодировка ASCII. Для указанных алфавитов в ней нет однозначного соответствия между кодом символа и его изображением. Поэтому национальные алфавиты было невозможно использовать для написания текстов программ, и они не входили в список терминальных символов. Современные системы программирования используют кодировку Unicode. Применение этого стандарта позволяет закодировать очень большое число символов из разных письменностей. В документах Unicode могут соседствовать китайские иероглифы, математические символы, буквы греческого алфавита, латиницы и кириллицы. Поэтому стало возможным, например, использование русскоязычных имен переменных и функций.

Часто к терминальным символам относят зарезервированные ключевые слова языка. Например, для языка C/C++ это слова ‘if’, ‘while’,

‘for’, ‘int’, ‘char’, ‘void’, ‘new’, ‘this’ и др. В языке Pascal – слова ‘DO’, ‘IF’, ‘THEN’, ‘VAR’, ‘BEGIN’, ‘END’ и т.д.

Определение 2.2. Нетерминальные символы – это символы, которые несут смысловое содержание фрагментов текста.

Пример: нетерминальный символ <функция> – он обозначает, что в этой части программы записана функция. Нетерминал <переменная> обозначает, что здесь будет имя переменной.

В учебниках по программированию часто можно встретить похожие записи:

```
while (Условие)
{
  БлокОпераций;
}
```

В данном случае это формат записи цикла с предусловием на языке C/C++. Вы знаете, что этот текст не будет компилироваться. Чтобы программа работала, необходимо заменить «Условие» на правильную запись, например «i<10». Также вместо «БлокОпераций» нужно написать те действия, которые будут выполняться на каждом шаге цикла. Например, «i=i+1».

Таким образом, нетерминальный символ – это такой символ, который обозначает некоторые важные части программы, и в дальнейшем будет заменен на последовательность терминальных символов.

Пример: Пусть есть нетерминал <УсловныйОператор>. Он будет заменяться по следующему правилу:

<УсловныйОператор> → if (<Условие>)
{
<БлокОпераций>;
}

Так мы задали формат записи условного оператора (ветвления) в нашей программе. Нетерминал <Условие> может быть записан разными способами. Например, так:

<Условие> → <Переменная> <ЗнакСравнения> <Переменная>

В свою очередь, имя переменной должно содержать только буквы и цифры и всегда начинаться с буквы. Это можно записать таким образом:

<Переменная> → <Буква> (<Буква> | <Цифра>)*

Сейчас эта запись кажется непонятной, но немного позже мы ее объясним.

Вертикальная черта отделяет взаимозаменяемые варианты в записи. Например, запись:

<Буква> → ‘a’ | ‘b’ | ‘c’ | ‘d’ | … | ’z’

Обозначает, что нетерминальный символ <Буква> может быть заменен на любую строчную букву латинского алфавита.

То же самое для цифр:

<Цифра> → ‘0’ | ‘1’ | ‘2’ | ‘3’ | … | ’9’

Знак сравнения можно задать так:

<ЗнакСравнения> → ‘>’ | ‘<’ | ‘==’ | ‘!=’ | ‘>=’ | ’<=’

В примерах и задачах мы часто будем обозначать нетерминалы печатными прописными буквами латинского алфавита (A, ..., Z) или записывать их в угловых скобках < >. Например, обозначения <нетерминал>, A, B, <Coshy>, <AB> являются нетерминальными символами. Обозначение AB следует читать как сочетание двух рядом стоящих нетерминалов A и B. От изменения обозначений нетерминальных символов грамматика не изменяется. Нетерминалы являются связующими элементами грамматики и поэтому их обозначение выбирается произвольно автором грамматики так, чтобы смысловая нотация грамматики была понятна и читабельна. Так, если разрабатывается грамматика числовой константы, то понятно, что одним из нетерминальных символов в этой грамматике будет

нетерминал <числовая константа>, который можно обозначить и сокращённо <ЧК>.

2.2 Словари

Определение 2.3. Словарь (алфавит) – это конечное непустое множество символов.

Словарь обозначается заглавными прописными буквами латинского алфавита (**A**, **B**, **C**, …), чаще – **V** и **S** (возможно с индексами). Элементы словаря перечисляются в фигурных скобках, поскольку словарь – это множество.

Термин алфавит как класс или группа символов введён для обозначения, например букв русского или латинского алфавитов соответственно. В теории формальных языков эта категория приняла лишь более строгую и конкретную форму.

Пример: Словарь **A** состоит из двух терминальных символов: a и b. Это записывается так: **A**={a, b}. Словарь **B**={0, 1} – бинарный словарь, состоит из двух цифр (0 и 1 – тоже терминальные символы). Стандарты ASCII и Unicode также являются примерами компьютерных алфавитов.

Отметим, что формальный словарь не обязательно должен состоять только из терминальных символов. В общем случае, как будет показано ниже, общий словарь **V** может включать набор как терминальных, так и нетерминальных символов.

2.3 Строки (цепочки символов)

Определение 2.4. Цепочка (строка) – это конечная последовательность символов. Обозначаются строки греческими прописными буквами – α, β, γ.

Пример: Строка α = «aa», строка β = «abc».

Определение 2.5. Длина цепочки – это число символов в цепочке.

Длина строки α обозначается так $|\alpha|$. Для приведённого примера длины цепочек соответственно равны $|\alpha| = 2$, $|\beta| = 3$.

Определение 2.6. Пустая строка – строка, которая не содержит символов. Ее можно обозначить так: ε = «». Или просто ε.Длина пустой цепочки ε равна 0: $|\varepsilon| = 0$.

2.4 Операции со строками

Определение 2.7. Конкатенация цепочек – это присоединение цепочек слева направо.

Конкатенация – это основная операция над строками. Все тексты на формальном языке составляются путем конкатенации отдельных цепочек по некоторым правилам.

Пример: Пусть α = «ab», β = «cab», $\gamma = \varepsilon$ (пустая строка). Тогда конкатенация строк α и β будет такой: $\alpha\beta$ = «abcab». Конкатенация строк α и γ: $\alpha\gamma$ = «abε» = «ab».

Свойства коммутативности над цепочками не выполняется в общем случае. То есть при перемене мест строк будет разный результат. Если α = «ab», β = «cab». Тогда $\alpha\beta$ = «abcab», а $\beta\alpha$ = «cabab».

Выделим ещё несколько операций над строками или цепочками символов.

Определение 2.8. Обращение – это запись символов цепочки α (обозначение α^R) в обратном порядке. Так, если α = «ab», то α^R = «ba».

Определение 2.9. Возведение строки в степень n – это операция n-кратной конкатенации символов цепочки. Обозначение: α^n.

Например, если α = «ab», то $\alpha^3 = \alpha\alpha\alpha$ = «ababab».

Для пустой цепочки ε справедливы следующие очевидные равенства:

- $\forall\alpha: \varepsilon\alpha = \alpha\varepsilon = \alpha$. Добавление к строке α пустой строки ε дает исходную строку α.
- $\varepsilon^R = \varepsilon$. При обращении пустой строки получаем пустую строку.

- $\forall n >= 0:\ \varepsilon^n = \varepsilon$. При возведении пустой строки в любую неотрицательную степень получаем пустую строку.
- $\forall \alpha:\ \alpha^0 = \varepsilon$. При возведении строки в нулевую степень получаем пустую строку.

Определение 2.10. Итерация строки – это множество строк, получаемых в результате возведения исходной строки в степень n = 0, 1, 2, 3,… Обозначение: $\alpha*$.

Например, если α = «ab», то $\alpha*$ = {α^0, α^1, α^2, α^3, …α^n, α^{n+1},…} = {ε, ab, abab, ababab, …$\underbrace{\text{ababab…ab}}_{\text{n раз}}$, …}.

Определение 2.11. Усеченная итерация строки – это множество строк, получаемых в результате возведения исходной строки в степень n = 1, 2, 3,… Обозначение: α^+.

Например, если α = «ab», то α^+ = {α^1, α^2, α^3, …α^n, α^{n+1},…} = {ab, abab, ababab, …$\underbrace{\text{ababab…ab}}_{\text{n раз}}$, …}.

Сравните определения итерации и усеченной итерации строки. Их отличия только в том, что в итерации степень строки начинается с нуля, а в усеченной итерации – с 1. Поэтому результат этих операций отличается только тем, что итерация порождает множество строк, включающее пустую строку. Можно записать: $\alpha* = \alpha^+ \cup \varepsilon$.

2.5 Операции со словарями

Определение 2.12. Произведение словарей A и **B** – это множество конкатенаций $\alpha\beta$ таких, что цепочка α принадлежит словарю **A**, а цепочка β принадлежит словарю **B**.

AB = {$\alpha\beta \mid \alpha \in A,\ \beta \in B$}.

Пример: Даны словари **A** = {a, b} и **B** = {c, d}. Тогда произведение словарей **AB** дает множество строк **AB** = {ac, ad, bc, bd}. А произведение **BA** дает другое множество цепочек: **BA** = {ca, cb, da, db}. Обратите внимание, что эти множества разные. Поэтому произведение словарей не коммутативно, **AB ≠ BA.**

Определение 2.13. Степень словаря. Для возведения словаря в степень n нужно найти произведение словаря на сам себя n раз.

Пусть **A** = {a, b, c}.

Для степени n=0: $A^0 = \{\varepsilon\}$. Нулевая степень словаря – это множество строк, состоящее только из пустой строки.

Для степени n=1: $A^1 = A = \{a, b, c\}$.

Для степени n=2: $A^2 = AA = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$.

Для степени n=3: $A^3 = AAA = A^2A = \{aaa, aab, aac, aba, abb, ...\}$.

Определение 2.14. Итерация словаря – это бесконечное объединение всех степеней словаря, *включая нулевую*. Обозначается A^*.

$A^* = A^0 \cup A^1 \cup A^2 \cup ... \cup A^n \cup ...$

Пример: A = {a, b}. Найти A^*.

Сначала найдем степени словаря:

$A^0 = \{\varepsilon\}$;

$A^1 = A = \{a, b\}$;

$A^2 = AA = \{aa, ab, ba, bb\}$;

$A^3 = A^2A = \{aaa, aba, baa, bba, aab, abb, bab, bbb\}$;

…

Теперь запишем объединения всех множеств:

$A^* = A^0 \cup A^1 \cup A^2 \cup ... \cup A^n \cup ... = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aba, baa, ...\}$.

В общем случае содержательный смысл итерации словаря включает множество всевозможных комбинаций всевозможной длины над элементами словаря.

Определение 2.15. Усеченная итерация словаря – это бесконечное объединение всех степеней словаря, *кроме нулевой*. Обозначается A^+.

Пример: Пусть **A** = {0, 1, 2,..., 9}, тогда $A^+ = \{0, 1, 2, \ldots 9, 10, 11, 12, \ldots 19, 20, \ldots 99, 100, \ldots\}$. Мы получили множество целых чисел.

Упражнения

1. Из фрагмента кода программы на языке C выпишите все терминальные символы:

```
#include <stdio.h>
int main(void)
{
    printf("Hello, world!\n");
    return 0;
}
```

2. Из фрагмента кода программы на языке Pascal выпишите все терминальные символы:

```
begin
writeln('Hello, world!');
end.
```

3. Из описания оператора switch языка C/C++ выпишите терминальные и нетерминальные символы:

```
switch (<выражение>)
{
 case <константное выражение 1>: <операторы 1>
 case <константное выражение 2>: <операторы 2>
 default: <операторы>
}
```

4. Из описания цикла for языка C/C++ выпишите терминальные и нетерминальные символы:

```
for (<начальное выражение>;
     <условное  выражение>;
     <выражение приращения>)
 <оператор>
```

5. Запишите словарь A, содержащий первый 5 символов английского алфавита.

6. Запишите словарь B, содержащий известные вам знаки арифметических операций.

7. Найдите длину цепочек:

α = «int»,

β = « »,

γ = «temp»,

δ = «=»,

φ = «2»,

ω = «+»,

τ = «3»,

μ = «;».

8. Найдите конкатенации строк из предыдущего примера: $\alpha\beta\gamma\mu$, $\gamma\delta\varphi\mu$, $\alpha\beta\gamma\delta\varphi\omega\tau\mu$.

9. Найдите итерации строк α, β и γ из примера 7.

10. Найдите усеченные итерации строк δ, φ и ω из примера 7.

11. Пусть **A**= {0, 1, 2}, **B** = {1, 0}. Найти $\mathbf{A}^3$, $\mathbf{B}^0$, **AB** и **BA**.

12. Пусть **V**={a, b, c}. Написать 8 самых коротких цепочек из A* и A^+.

Список использованных источников

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