Глава 1. Элементы теории формальных языков и грамматик

1.1. Основные понятия и определения

Язык – это множество *строк* (*предложений*, *цепочек*). Каждая строка языка формируется из словаря в соответствии с заданными правилами.

Строка есть конечная последовательность символов, каждый из которых принадлежит некоторому конечному *алфавиту* (*словарю языка*) V; при этом символы в строке могут повторяться. Если строка содержит m символов, то говорят, что она имеет длину m. Строка длины 0, т.е. не содержащая ни одного символа, называется *пустой строкой* и обозначается ε. Длина строки x обозначается $|x|$.

Пусть V – некоторый алфавит. Тогда через V^* (рефлексивно-транзитивное замыкание V) обозначается множество всех строк (включая пустую строку), составленных из символов, входящих в V, т. е. это множество определенных над алфавитом V строк. Через V^+ обозначается множество всех строк, исключая пустую строку, т. е. $V^* = V^+ \cup \{\varepsilon\}$.

Пример: $V = \{0, 1\}$.

$V^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, ...\}$.

$V^+ = \{0, 1, 00, 01, 10, 11, 000, ...\}$.

Если $\alpha \in V^*$ – строка длины m, а $\beta \in V^*$ – строка длины n, то их объединение, обозначаемое $\alpha\beta$, есть конкатенация (сцепление) строк α и β. В строке $\alpha\beta$ подстроку α называют *префиксом* строки $\alpha\beta$, а подстроку β – *суффиксом* строки $\alpha\beta$. В результате объединения получается строка длины $m + n$, т. е. $|\alpha\beta| = m + n$. Например, если $\alpha = a_1a_2...a_m$, а $\beta = b_1b_2...b_n$, тогда $\alpha\beta = a_1a_2...a_mb_1b_2...b_n$. Объединение является ассоциативной операцией, т. е. $(\alpha\beta)\gamma = \alpha(\beta\gamma) = \alpha\beta\gamma$, но не является коммутативной операцией, т. к. в общем случае $\alpha\beta \neq \beta\alpha$. Для пустой строки, очевидно, справедливы следующие утверждения:

$\varepsilon\alpha = \alpha\varepsilon = \alpha$ для любого $\alpha \in V^*$.

Часто используется обозначение α^n (n – степень строки α), соответствующее конкатенации n строк α: $\alpha^n = \underbrace{\alpha\alpha\ldots\alpha\alpha\alpha}_{n}$. Свойства степени: $\alpha^0 = \varepsilon$; $\alpha^n = \alpha\alpha^{n-1} = \alpha^{n-1}\alpha$.

Обычно совокупность строк, принадлежащих V^* и имеющих длину 2, обозначают V^2, имеющих длину 3 – соответственно V^3 и т. д.; V^0 – пустая строка. Тогда

$$V^+ = \bigcup_{i=1}^{\infty} V^i \text{ и } V^* = V^+ \cup \{\varepsilon\} = \bigcup_{i=0}^{\infty} V^i .$$

Формальным языком L над алфавитом V называется произвольное подмножество множества V^*. Если L_1, L_2 – два формальных языка, то их объединение $L_1L_2 = \{\alpha\beta \mid \alpha \in L_1, \beta \in L_2\}$ также является формальным языком. Например, если $L_1 = \{11, 1\}$ и $L_2 = \{\varepsilon, a, bb\}$, то $L_1L_2 = \{11, 1, 11a, 1a, 11bb, 1bb\}$.

Объединение формального языка L с самим собой записывается как L^2, или, в общем случае, как

$$L^0 = \{\varepsilon\},\ L^1 = L,\ L^i = LL^{i-1} = L^{i-1}L \text{ для } i \geq 2.$$

Синтаксис просого языка можно формально описать, используя систему представления множеств. Например, язык $L = \{a^n b^n \mid n \geq 0\}$ включает строки, состоящие из n символов a и последующих n символов b. Пустая строка (при $n = 0$) также принадлежит языку.

Синтаксис сложного языка (к ним относятся и языки программирования) описывается с помощью *грамматики*. Грамматика состоит из набора правил для получения (порождения) строк языка.

1.2. Метаязыки

Язык, используемый для описания синтаксиса какого-либо языка, называется *метаязыком*. Для формального описания синтаксиса языков программирования (*формальных языков*) большое распространение получили такие метаязыки, как *синтаксические диаграммы* (СД) и *формы Бэкуса-Наура* (БНФ).

Рассмотрим диаграмму на рис. 1.1, иллюстрирующую понятие «условный оператор». Прямоугольные вершины соответствуют *нетерминальным символам* (или *нетерминалам*) языка, которые должны быть представлены отдельными СД, определяющими соответствующее понятие. В кружках (овалах) указаны *терминальные символы* (или *терминалы*) языка, входящие в словарь языка (ключевые слова, символы-разделители и т. п.). Ориентированный путь по дугам от начальной вершины до конечной вершины определяет порядок формирования синтаксического понятия.

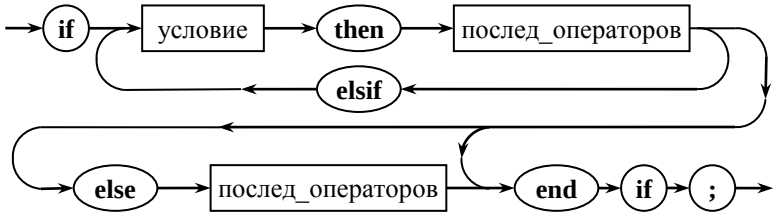

Рис. 1.1. Пример синтаксической диаграммы

Альтернативой синтаксическим диаграммам являются формы Бэкуса-Наура. БНФ состоит из набора *правил* (или *продукций*). Нетерминальные символы языка (металингвистические переменные) заключаются в угловые скобки вида < и >. Правило БНФ состоит из левой и правой частей, разделенных метасимволом «::=». Метасимвол «::=» означает «определяется как», «это», «есть». В левой части правила записывается нетерминал (название определяемого синтаксического понятия). Правая часть определяет соответствующее синтаксическое понятие. При наличии альтернативных определений они разделяются метасимволом «|», означающим «или».

Представим одно из возможных определений синтаксического понятия «условный оператор» в форме БНФ (нетерминал <пусто> означает пустую строку):

<условный оператор> ::= **if** <условие>
    then <послед_операторов> <else-часть> **end if ;**
<else-часть> ::= <пусто>
    | <список_elsif> **else** <послед_операторов>
<список_elsif> ::= <пусто> **|** <элемент_elsif> <список_elsif>
<элемент_elsif> ::= **elsif** <условие>
    then <послед_операторов>

Для БНФ характерно наличие рекурсивных правил (в нашем примере определение нетерминала <список_elsif>, где этот нетерминал встречается как в левой, так и в правой части). Такие правила обычно определяют различные списковые конструкции.

Процесс построения синтаксической конструкции заключается в выводе этой конструкции путем последовательных подстановок правых частей правил БНФ вместо нетерминалов из левых частей. Такой процесс продолжается до тех пор, пока не будет получена конструкция, состоящая только из терминальных символов.

На практике для удобства применяют расширенные формы Бэкуса-Наура (РБНФ). Существует множество различных вариантов РБНФ. Рассмотрим один из таких вариантов (он достаточно близок к международному стандарту РБНФ ISO/IEC 14977).

Металингвистическая переменная (нетерминал) обозначается произвольной символьной строкой (без использования угловых скобок как в БНФ).

Левая и правая части правила разделяются метасимволом "=" (вместо "::=" в БНФ), альтернативные варианты разделяются метасимволом "|". Каждое правило заканчивается метасимволом «;» (точка с запятой), альтернативой символу «;» является символ «.» (точка).

Конкатенация определяется метасимволом «,» (запятая). Правило вида A = B, C. означает, что нетерминал A состоит из двух символов – B и C. Элементы конкатенации называют ещё *синтаксическими факторами*, или просто *факторами*. В таком случае несколько слов, написанных через пробелы, следует понимать как одно многословное имя нетерминального символа (пробелы не являются разделителями символов). Если же не использовать в качестве конкатенации символ «,», а сохранить пробел в качестве разделителя символов, тогда если нетерминал состоит из нескольких смысловых слов, то они записываются слитно или разделяются символом подчеркивания или дефисом. При слитном написании для удобства восприятия целесообразно каждое ее слово начинать с прописной буквы. Например, нетерминал, означающий список идентификаторов, можно записать как «СписокИдентификаторов», «Список-идентификаторов», или «Список_идентификаторов».

Терминальные символы изображаются словами, написанными буквами латинского алфавита (ключевые слова) или цепочками символов, заключенными в одиночные (′) или двойные (″) кавычки. Для удобства восприятия ключевые слова дополнительно можно выделить жирным шрифтом.

Условное вхождение. Квадратные скобки «[» и «]» означают, что заключенная в них синтаксическая конструкция может отсутствовать.

Повторение. Фигурные скобки «{» и «}» означают нуль или более повторений заключенной в них синтаксической конструкции.

Наиболее важными расширениями являются условное вхождение и повторение. Остальные расширения связаны в основном с синтаксисом обозначений метасимволов.

Ниже приведен пример определений синтаксического понятия «условный оператор» в форме РБНФ.

Условный оператор = "**if**", Условие, "**then**",
    Последовательность операторов,
    { "**elsif**", Условие, "**then**",
    Последовательность операторов },
    ["**else**", Последовательность операторов],
    "**end**", "**if**", "**;**" ;

Если не использовать запятую в качестве операции конкатенации, то из многословных обозначений нетерминалов следует убрать пробелы, чтобы обозначения воспринимались как единое целое, например, следующим образом:

УсловныйОператор = "**if**" Условие "**then**"
    Последовательность-операторов
    { "**elsif**" Условие "**then**"
    Последовательность-операторов }
    ["**else**" Последовательность-операторов]
    "**end**" "**if**" "**;**" ;

Как видно из примеров, определения в РБНФ получаются более простыми, чем в БНФ, отсутствуют рекурсивные правила, не требуется введение новых нетерминалов.

Нотация БНФ и РБНФ эквивалентна синтаксическим диаграммам и предназначена для описания синтаксиса контекстно-свободных языков программирования. Всегда можно перейти от одного способа описания к другому.

Недостатком рассмотренных метаязыков является то, что они описывают грамматическую структуру формального языка без учёта контекстных зависимостей. При наличии таких зависимостей описание оказывается неполным, и некоторые правила синтаксиса описываемого языка приходится излагать в обычной текстовой форме. Например, в ряде типизированных языков требуется описание типа всех используемых в программе переменных. Сформулировать это требование с помощью нотации БНФ (РБНФ) или СД невозможно.

Пример:

1. Программа = "**main**" Идент "**;**" Блок "**endmain**".
2. Идент = Буква { Буква | Цифра }.
3. Блок = { Описание } "**begin**" ПоследОператоров "**end**".
4. Описание = "**type**" ОписаниеТипа "**;**" | "**var**" ОписаниеПерем "**;**".
5. ОписаниеТипа = Идент "**is**" Тип.
6. ОписаниеПерем = СписокИдент "**:**" Тип.
7. СписокИдент = Идент { "**,**" Идент }.
8. Тип = ПростойТип | ТипМассив.
9. ПростойТип = Идент.
10. ТипМассив = "**array**" "**[**" ЧисловаяКонст "**]**" "**of**" ПростойТип.
11. ПоследОператоров = Оператор { "**;**" Оператор }.
12. Оператор = Присваивание | Цикл.
13. Присваивание = Переменная "**:=**" Выражение.
14. Переменная = Идент | ИндексПеременная.
15. ИндексПеременная = Идент "**[**" ПростоеВыраж "**]**".

16. Цикл = "**while**" Выражение "**do**" ПоследОператоров "**enddo**".
17. Выражение = ПростоеВыраж [Отношение ПростоеВыраж].
18. Отношение = "<" | "<=" | ">" | ">=" | "=" | "#".
19. ПростоеВыраж = Терм { АддитОперация Терм }.
20. АддитОперация = "+" | "–" | "**or**".
21. Терм = Фактор { МультОперация Фактор }.
22. МультОперация = "*****" | "/" | "**and**".
23. Фактор = Константа | Переменная | "**(**" Выражение "**)**" | "**not**" Фактор.
24. Константа = ЧисловаяКонст | ЛогическаяКонст.
25. ЧисловаяКонст = Целое ["**.**" Целое].
26. Целое = Цифра { Цифра }.
27. Цифра = "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9".
28. ЛогическаяКонст = "**true**" | "**false**".

Определение нетерминала «Буква» здесь не приведено ввиду его очевидности – определяется выбранным алфавитом (обычно строчные и прописные буквы латинского алфавита).

Краткая характеристика языка и семантические соглашения:

- Язык удовлетворяет семантическим соглашениям, характерным для многих языков программирования (единственность именования различных объектов программы, необходимость описания идентификатора до его использования и т.п.).
- В ключевых словах и идентификаторах прописные и строчные буквы не различаются.
- Ключевые слова языка зарезервированы, их нельзя использовать в качестве идентификаторов.
- Отсутствуют именованные константы.
- Отсутствуют унарные операции + и –, т.е. для представления, например, отрицательного числа –8, надо использовать бинарную операцию 0 – 8.
- Предопределенные типы: integer, float, Boolean.
- В соответствии с правилом 10 допускаются только одномерные массивы с диапазоном индексов от 1 до «ЧисловаяКонст», т.е. индексация элементов всегда начинается с 1, а «ЧисловаяКонст» указывает размер массива и соответствует индексу последнего элемента и может быть только целого типа. Если в конструкции в качестве «ПростойТип» используется идентификатор («Идент»), он должен представлять именно простой тип, а не массив.
- В конструкции «ИндексПеременная» (правило 15) выражение, используемое в качестве индекса, должно быть целого типа.
- В операторе цикла конструкция «Выражение» после ключевого слова **while** должно быть типа Boolean (правило 16).
- Комментарий представляет собой любую последовательность символов, заключенную в фигурные скобки "{" и "}".

В языке (в соответствии с приведенной выше грамматикой) определена следующая приоритетность арифметических и логических операций (в порядке убывания приоритетов):

1. Операция "**not**"
2. Мультипликативные операции: "*****" | "/" | "**and**"
3. Аддитивные операции: "+" | "–" | "**or**"
4. Операции отношения: "<" | "<=" | ">" | ">=" | "=" | "#"