Глава 1

ЯЗЫКИ ИХ ПРЕДСТАВЛЕНИЕ

§ 1.1. Алфавиты и языки

Приступая к изучению теории языков, прежде всего следует определить, что мы подразумеваем под термином язык. Энциклопедическое определение языка как "важнейшего средства общения, обмена мыслями и взаимного понимания в человеческом обществе"[1] недостаточно точно для построения математической теории языков. Поэтому мы определим язык абстрактно — как математическую систему. Эта формальность даст нам возможность делать строгие утверждения о формальных языках, которые надлежащим образом будут моделироваться. С этой мыслью мы дадим следующие определения.

***Определение* 1.1.** *Алфавит* или *словарь* есть конечное множество символов.

Что такое *символ* — не определяется, как не определяется, например, точка в геометрии.

Некоторые примеры алфавитов: *латинский* $\{A, B, ..., Z\}$; *греческий* $\{\alpha, \beta, ..., \omega\}$; *бинарный* $\{0, 1\}$.

***Определение* 1.2.** *Предложение* (*строка*, *слово*) есть любая цепочка конечной длины, составленная из символов алфавита. Предложение, не содержащее ни одного символа, называется *пустым предложением*. Оно обозначается греческой буквой ε.

Если V — некоторый алфавит, то V^* обозначает множество всех предложений, составленных из символов алфавита V, включая и пустое предложение.

Будем обозначать посредством V^+ множество $V^* \setminus \{\varepsilon\}$. Так, например, если $V = \{0, 1\}$, то $V^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, ...\}$, а $V^+ = \{0, 1, 00, 01, 10, 11, ...\}$.

Если $x \in V^*$, то $|x|$ обозначает *длину* цепочки x, т. е. число символов, ее составляющих. Естественно считать, что длина пустой цепочки ε равна 0.

***Определение* 1.3.** *Язык* есть любое множество предложений над некоторым алфавитом. Формально, если V — *алфавит*, L — *язык*, то $L \subseteq V^*$.

Естественно возникают три вопроса:

1. Как представить язык, т. е. как определить предложения, составляющие язык?
2. Для каждого ли языка существует конечное представление?
3. Какова структура тех классов языков, для которых существуют конечные представления?

[1] Энциклопедический словарь. — М.: Большая советская энциклопедия, 1955. С. 716.

На первый вопрос ответить легко, если язык конечен. Надо просто перечислить все предложения, т. е. составить список. Если язык бесконечен, то возникает проблема нахождения способа *конечного представления* бесконечного языка. Это конечное представление само будет строкой символов над некоторым алфавитом с подразумеваемой интерпретацией, которая связывает это конкретное представление с данным языком.

На второй вопрос ответ отрицателен. В следующем параграфе будет показано, что множество всех предложений над данным словарем счетно-бесконечно.

То, что множество всех подмножеств счетно-бесконечного множества не является счетно-бесконечным, есть хорошо известный факт теории множеств[2]. Другими словами, множество всех языков над данным алфавитом не счетно. Ясно, что конечное представление языка будет предложением какого-то (обычно, другого) языка (*метаязыка*). Но этот последний, как и любой другой язык, не более чем счетно-бесконечное множество предложений. Так что существует значительно больше языков, чем конечных представлений.

Ответу на третий вопрос посвящена бóльшая часть дальнейшего изложения.

§ 1.2. Представление языков

Распознавание. Один из способов представления языка состоит в том, чтобы дать *алгоритм*[3], который определяет, есть ли данное предложение в данном языке или нет.

Более общий способ — дать *процедуру*, которая прекращает работу с ответом “да” для предложений в языке и либо не завершается[4], либо завершается с ответом “нет” для предложений не из языка. Говорят, что такие алгоритм и процедура *распознают* язык.

Существуют языки, которые можно распознать при помощи процедуры, но не при помощи алгоритма.

Порождение. Другой способ представления языка — дать процедуру, которая систематически порождает предложения языка последовательно в некотором порядке.

Если имеется алгоритм или процедура, которые распознают предложения языка над алфавитом V, то на их основе можно построить порождающий механизм — алгоритм или процедуру, порождающий этот язык. Именно, можно систематически генерировать все предложения из множества V^*, проверять каждое

---

[2] Теорема Г. Кантора (1878 г.).

[3] Согласно Д. Кнуту [11] *алгоритм* — это свод конечного числа правил, задающих последовательность выполнения операций при решении той или иной конкретной задачи. Для алгоритма характерны следующие пять особенностей: (1) *конечность* — завершаемость после выполнения конечного числа шагов; (2) *определенность* — однозначность; (3) *ввод* — исходные данные; (4) *вывод* — результат; (5) *эффективность* — выполнимость любой операции за конечное время. В вырожденных случаях ввод и (или) вывод могут отсутствовать.

[4] Этим она и отличается от алгоритма.

предложение на принадлежность его языку с помощью распознающего механизма и выводить в список только предложения языка. Однако если используется распознающая процедура, а не алгоритм, то есть опасность, что процесс порождения никогда не продвинется дальше первого же предложения, на котором процедура не завершается.

Для предотвращения этой опасности надо организовать проверку таким образом, чтобы распознающая процедура никогда не продолжала проверку одного предложения вечно. Сделать это можно следующим образом.

Пусть в алфавите V имеется p символов. Предложения из множества V^* можно рассматривать как числа p-ичной системы счисления плюс пустое предложение ε. Пронумеруем предложения из V^* в порядке увеличения длины, причем все предложения одинаковой длины будем нумеровать в "числовом" порядке. Например, если $V = \{a, b, c\}$, то предложения из V^* будут занумерованы следующим образом:

1.	ε	5.	*aa*	9.	*bb*
2.	*a*	6.	*ab*	10.	*bc*
3.	*b*	7.	*ac*	11.	*ca*
4.	*c*	8.	*ba*	12.	...

Тем самым показано, что множество предложений над алфавитом V счетно.

Пусть P — распознающая процедура для языка L. Предполагается, что она разбита на шаги так, что имеет смысл говорить об i-м шаге этой процедуры для любого заданного предложения.

Прежде чем дать процедуру перечисления предложений языка L, определим способ нумерации пар положительных целых. Можно отобразить все множество пар положительных целых (i, j) на множество положительных целых, как показано в табл. 1.1, при помощи следующей формулы:

$$k = \frac{(i + j - 1)(i + j - 2)}{2} + j.$$

Табл. 1.1

i \ j	1	2	3	4	5
1	1	3	6	10	15
2	2	5	9	14	...
3	4	8	13	...	
4	7	12	...		
5	11	...			

Замечание 1.1. Поясним, как получена вышеприведенная формула. Пусть имеется некоторое k, занимающее в сетке позицию (i, j). Ясно, что $k = N + j$, здесь N — число элементов внутри треугольной сетки с основанием, концы которого имеют координаты $(i, 1)$ и $(1, i)$. Очевидно, что

$$N = 1 + 2 + 3 + ... + n = \frac{n(n + 1)}{2},$$

здесь n — число рядов клеток треугольной сетки, параллельных ее основанию, считая и само это основание. Очевидно также, что $i+j=n+2$. Следовательно, $n=i+j-2$. Подставив последнее выражение для n в формулу для N, получим приведенное ранее выражение для k.

Итак, мы можем перенумеровать упорядоченные пары целых чисел согласно присвоенному номеру k. Несколько первых пар есть: (1, 1), (2, 1), (1, 2), (3, 1), (2, 2), ... Любая заданная пара чисел (i , j) действительно окажется в списке под номером

$$k = \frac{(i+j-1)(i+j-2)}{2} + j.$$

Теперь мы можем описать процедуру перечисления, т. е. *порождающую процедуру*, предложений языка L. Процедура перенумеровывает пары целых в соответствии с табл. 1.1. Когда пара (i , j) занумеровывается, процедура порождает i-е предложение из множества V^* и выполняет первые j шагов распознающей процедуры P над этим предложением. Когда процедура P определяет, что порожденное предложение принадлежит языку L, порождающая процедура добавляет это предложение к списку предложений языка L. Теперь, если слово номер i принадлежит языку L, то этот факт устанавливается при помощи процедуры P за j шагов для некоторого конечного j. Очевидно, что таким образом организованная процедура будет перечислять все предложения языка L.

Однако если имеется порождающая процедура для языка L, то на ее основе можно построить *распознающую процедуру* для этого языка. Действительно, чтобы определить, имеется ли предложение x в L, достаточно с помощью порождающей процедуры последовательно порождать предложения языка L и сравнивать каждое с x. Если x порождается, то распознающая процедура заканчивается, распознав x. Конечно, если предложение x не в L, то распознающая процедура никогда не закончится.

***Определение* 1.4.** *Я*зык, предложения которого могут порождаться процедурой, называется *рекурсивно перечислимым*. Также принято говорить, что язык рекурсивно перечислим, если существует процедура распознавания предложений этого языка.

***Определение* 1.5.** *Я*зык *рекурсивен*, если существует алгоритм его распознавания.

Известно, что класс рекурсивных языков является собственным подмножеством класса рекурсивно перечислимых языков. Более того, существуют языки и похуже, которые даже не рекурсивно перечислимы, т.е. невозможно эффективно перечислять предложения такого языка. Пример языка, не являющегося рекурсивно перечислимым множеством приводится в начале §7.3.

Теорема 1.1. *Пусть $L \subseteq V^*$ — некоторый язык, а $\overline{L} = V^* \setminus L$ — его дополнение. Если языки L и $\overline{L}$ рекурсивно перечислимы, то язык L рекурсивен.*

Доказательство. Пусть язык L распознается процедурой P, а его дополнение $\overline{L}$ распознается процедурой $\overline{P}$.

Построим алгоритм распознавания языка L, т.е. отвечающий на вопрос: $x \in L$?, следующим образом:

Шаг 1. $i := 1$.

Шаг 2. Применим i шагов процедуры P к цепочке x. Если процедура P не завершается, то перейти к шагу 3, иначе — к шагу 4.

Шаг 3. Применим i шагов процедуры $\overline{P}$ к цепочке x. Если процедура $\overline{P}$ не завершается, то $i := i + 1$ и перейти к шагу 2.

Шаг 4. При некотором i одна из этих двух процедур завершится, распознав цепочку x, так как либо $x \in L$ и это распознает процедура P, либо $x \notin L$ и это распознает процедура $\overline{P}$. Соответственно распознающий алгоритм выдает либо положительный, либо отрицательный ответ.

Поскольку язык L распознается описанным алгоритмом, то он *рекурсивен*. Что и требовалось доказать.

Завершая эту главу, можем сказать, что теория языков изучает множества строк символов (предложений), их представление, структуру и свойства.