лекция №2

Языка и их представление

Алфавиты, цепочки и языка

Алфавит, или словарь - это конечное множество символов. для обозначения символов мы будем пользоваться цифрами, латинскими буквами и специальными буквами типа #, $.

Пусть V - алфавит. Цепочка в алфавите V - это любая строка конечной длины, составленная из символов алфавита V. Синонимом цепочки является предложение, строка и слово. Пустая строка (обозначается e) - это цепочка, в которую входит ни один символ.

Конкатенацией цепочек x и y называется цепочка xy. Заметим, что xe = ex = x для любого цепочки x.

Пусть x, y, z - произвольные цепочки в некотором алфавите. цепочка y называется подцепочкой цепочки xyz. Цепочки x и y называются, соответственно, префиксом и суффиксом цепочки xy. Заметим, что любой префикс или суффикс цепочки является подцепочкой этой цепочки. Кроме того, пустой цепочка является префиксом, суффиксом и подцепочкой для любого цепочки.

Пример 2.1. Для цепочки abbba префиксом является любая цепочка из множеств L1 = {e, a, ab, abb, abbb, abbba} суффиксом является любая цепочка из множеств L2 = {e, a, ba, bba, bbba, abbba} пидланцюжком является любая цепочка с множеств L1 L2.

Длиной цепочки w (обозначается | w |) называется число символов в ней. Например, | abababa | = 7, a | e | = 0.

Язык в алфавите V - это некоторое множество цепочек в алфавите V.

Пример 2.2. Пусть дано алфавит V = {a, b}. Вот некоторые языки в алфавите V:

L1 = - пустой язык;

L2 = {e} - язык, содержащий только пустую цепочку

(Заметим, что L1 и L2 - разные языки)

L3 = {e, a, b, aa, ab, ba, bb} - язык, содержащий цепочки с a и b, длина которых не превышает 2;

L4 - язык, включая всевозможные цепочки с a и b, содержащих четное число a и четное число b;

L5 = {an2 | n> 0} - язык цепочек из a, длины которых представляют собой квадраты натуральных чисел.

Два последних языка содержат бесконечное число цепочек.

Введем обозначения V * для множеств всех цепочек в алфавите V, включая пустую цепочку. Каждый язык в алфавите V является подмножеством V * . Для обозначения множеств всех цепочек в алфавите V, кроме пустого цепочки, будем использовать V +.

Пример 2.3. Пусть V = {0, 1}. Тогда V * = {e, 0, 1, 00, 01, 10, 11, 000, …}, V + = {0, 1, 00, 01, 10, 11, 000, ...}.

Введем некоторые операции над языками.

Пусть L1 и L2 - языка в алфавите V. конкатенацию языков L1 и L2 называется речь L1L2 = {xy | x L1, y L2}.

Пусть L - язык в алфавите V. итерации языка L называется язык L * который определяется следующим образом:

L0 = {e};

Ln = LLn-1, n 1,

L * = n = 0Ln.

Пример 2.4. Пусть L1 = {aa, bb} и L2 = {e, a, bb}. тогда

L1L2 = {aa, bb, aaa, bba, aabb, bbbb}, и

L1 * = {e, aa, bb, aaaa, aabb, bbaa, bbbb, aaaaaa ...}.

Большинство языков, представляющих интерес, содержащие бесконечное число цепочек. При этом возникают три важных вопроса.

Во-первых, как представить язык (т.е. специфицировать входящие в него цепочки)? Если язык содержит только конечное множество цепочек, ответ прост. Можно просто перечислить его цепочки. если речь бесконечна, необходимо найти для нее конечное представление. это конечное

представления, в свою очередь, будет строкой символов над некоторыми алфавиту вместе с некоторой интерпретацией, что связывает это представление с языком.

Во-вторых, для любого ли языка существует конечное представление? можно предположить, что ответ отрицательный. Мы увидим, что множество всех цепочек над алфавитом счетно. Язык - это любое подмножество цепочек. Из теории множеств известно, что множество всех подмножеств счетного множеств неисчислимо. Хотя мы и не дали строгого определения того, что является конечным представлением, интуитивно понятно, что любое разумное определение конечного представления ведет только к счетного множеств конечных представлений, поскольку нужно иметь возможность записать такое конечное представление в виде строки символов конечной длины. Поэтому как значительно больше, чем конечных представлений.

В-третьих, можно спросить, какова структура тех классов языков, для которых существует конечное представление?

Представление языка.

Процедура - это конечная последовательность инструкций, которые могут быть механически выполнены. Примером может служить машинная программа. Процедура, которая всегда заканчивается, называется алгоритмом.

Один из способов представления языка - дать алгоритм, определяющий, принадлежит цепочка языка. Более общий способ состоит в том, чтобы дать процедуру, которая останавливается с ответом «да» для цепочек, принадлежащих языке, и либо останавливается с ответом «нет», или вообще не останавливается для цепочек, которые не относятся языке. Говорят, что такая процедура или метод распознает язык.

Такой метод представляет язык с точки зрения распознавания. Язык можно также представить методом порождения. В частности, можно дать процедуру, систематически порождает в определенном порядке цепочки языка.

Если мы можем распознать цепочки языка над алфавитом V или помощью процедуры, или с помощью алгоритма, то мы можем и генерировать язык, поскольку мы можем систематически генерировать все

цепочки с V *, проверять каждую цепочку на принадлежность языка и выдавать список только цепочек языка. Но если процедура не всегда заканчивается при проверке цепочки, мы не сдвинемся дальше первой цепочки, на которой процедура не заканчивается. Эту проблему можно обойти, организовав проверку таким образом, чтобы процедура никогда не продолжала проверять одну цепочку бесконечно. Для этого введем следующую конструкцию.

Предположим, что V имеет p символов. Мы можем рассматривать цепочки с V * как числа, представленные в базисе p, плюс пустой цепочку e. можно занумеровать цепочки в порядке возрастания длины и в «числовом» порядке для цепочек одинаковой длины. Такая нумерация для цепочек языка {a, b, c} * приведена на рис. 2.1, а.

Пусть P - процедура для проверки принадлежности цепочки языка L. Предположим, что P может быть представлена дискретными шагами, так что имеет смысл говорить о i-м шаге процедуры для любого данного цепочки. Прежде чем дать процедуру перечисления цепочек языка L, дадим процедуру нумерации пар положительных чисел.

Все упорядоченные пары положительных чисел (x, y) можно отобразить на множество положительных чисел следующей формуле:

Z = (x + y-1) (x + y-2) / 2 + y

Пары целых положительных чисел можно упорядочить в соответствии со значением z (рис. 2.1, б).

1	e
2	a
3	b
4	c
5	aa
6	ab
7	ac
8	ba
9	bb
...	...

а

y

x	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			$z(x, y)$	

б

Рисунок 2.1

Теперь можно дать процедуру перечисления цепочек L. нумеруя упорядоченные пары целых положительных чисел - (1,1), (2,1), (1,2), (3,1), (2,2), ... При нумерации пары (i, j) генерируем i-й цепочку с V * и применяем к цепочке первой j шагов процедуры P. Как только мы определили, что сгенерировала цепочка принадлежит L, добавляем цепочку в список элементов L. Если цепочка i принадлежит L, это будет определено P по j шагов для некоторого конечного j. При перечислении (i, j) будет сгенерирована цепочку с номером i. Легко видеть, что эта процедура перечисляет все цепочки L.

Если мы процедуру генерации цепочек языка, то мы всегда можем построить процедуру распознавания предложений языка, но не всегда алгоритм. Для определения того, принадлежит ли x языке L, просто нумеруя предложения L и сравниваем x с каждым предложением. Если сгенерирован x, процедура останавливается, распознав, что x принадлежит L. Конечно, если x НЕ принадлежит L, процедура никогда не закончится.

Язык, предложения которой могут быть сгенерированы процедурой, называется рекурсивно перечисляемых. Язык рекурсивно перечислим, если имеется процедура, распознает предложения языка. Говорят, что речь рекурсивен, если существует алгоритм для распознавания речи. класс рекурсивных как является собственным подмножеством класса рекурсивно перечисляемых языков. Мало того, существуют языки, которые не являются даже рекурсивно перечисляемых.