Лекція №2

Мови і їх уявлення

Алфавіти, ланцюжки та мови

Алфавіт, або словник - це кінцева множина символів. Для позначення символів ми будемо користуватися цифрами, латинськими літерами та спеціальними літерами типу #, $.

Нехай V - алфавіт. Ланцюжок в алфавіті V - це будь-який рядок кінцевої довжини, складена з символів алфавіту V. Синонімом ланцюжка є пропозиція, рядок і слово. Порожній ланцюжок (позначається e) - це ланцюжок, в яку не входить жоден символ.

Конкатенацією ланцюжків x і y називається ланцюжок xy. Зауважимо, що xe = ex = x для будь ланцюжка x.

Нехай x, y, z - довільні ланцюжки в деякому алфавіті. Ланцюжок y називається подцепочкой ланцюжка xyz. Ланцюжки x і y називаються, відповідно, префіксом і суфіксом ланцюжка xy. Зауважимо, що будь префікс або суфікс ланцюжка є подцепочкой цього ланцюжка. Крім того, порожній ланцюжок є префіксом, суфіксом і подцепочкой для будь ланцюжка.

Приклад 2.1. Для ланцюжка abbba префіксом є будь-яка ланцюжок з множинаі L1 = {e, a, ab, abb, abbb, abbba}, суфіксом є будь-яка ланцюжок з множинаі L2 = {e, a, ba, bba, bbba, abbba}, підланцюжком є будь-яка ланцюжок з множинаі L1 L2.

Довжиною ланцюжка w (позначається | w |) називається число символів в ній. Наприклад, | abababa | = 7, a | e | = 0.

Мова в алфавіті V - це деяка множина ланцюжків в алфавіті V.

Приклад 2.2. Нехай дано алфавіт V = {a, b}. Ось деякі мови в алфавіті V:

L1 = - порожній мову;

L2 = {e} - мова, що містить тільки порожню ланцюжок

(Зауважимо, що L1 і L2 - різні мови);

L3 = {e, a, b, aa, ab, ba, bb} - мова, що містить ланцюжки з a і b, довжина яких не перевершує 2;

L4 - мова, що включає всілякі ланцюжки з a і b, що містять парне число a і парне число b;

L5 = {an2 | n> 0} - мову ланцюжків з a, довжини яких представляють собою квадрати натуральних чисел.

Два останніх мови містять нескінченне число ланцюжків.

Введемо позначення V * для множинаі всіх ланцюжків в алфавіті V, включаючи порожню ланцюжок. Кожна мова в алфавіті V є підмножиною V *. Для позначення множинаі всіх ланцюжків в алфавіті V, крім порожнього ланцюжка, будемо використовувати V +.

Приклад 2.3. Нехай V = {0, 1}. Тоді V * = {e, 0, 1, 00, 01, 10, 11, 000, ...}, V + = {0, 1, 00, 01, 10, 11, 000, ...}.

Введемо деякі операції над мовами.

Нехай L1 і L2 - мови в алфавіті V. Конкатенацією мов L1 і L2 називається мова L1L2 = {xy | x L1, y L2}.

Нехай L - мову в алфавіті V. Ітерацією мови L називається мова L *, який визначається наступним чином:

L0 = {e};

Ln = LLn-1, n 1;

L * = n = 0Ln.

Приклад 2.4. Нехай L1 = {aa, bb} і L2 = {e, a, bb}. Тоді

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

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

Більшість мов, що представляють інтерес, містять нескінченне число ланцюжків. При цьому виникають три важливих питання.

По-перше, як представити мову (тобто специфікувати вхідні в нього ланцюжка)? Якщо мова містить тільки кінцеве множина ланцюжків, відповідь проста. Можна просто перерахувати його ланцюжка. Якщо мова нескінченна, необхідно знайти для неї кінцеве уявлення. Це кінцеве

представлення, в свою чергу, буде рядком символів над деякими алфавітом разом з деякою інтерпретацією, що зв'язує це подання з мовою.

По-друге, для будь-якого Чи мови існує кінцеве уявлення? Можна припустити, що відповідь негативна. Ми побачимо, що множина всіх ланцюжків над алфавітом лічильно. Мова - це будь-яке підмножина ланцюжків. З теорії множин відомо, що множина всіх підмножин рахункового множинаі незліченно. Хоча ми і не дали строгого визначення того, що є кінцевим представленням, інтуїтивно зрозуміло, що будь-яке розумне визначення кінцевого уявлення веде тільки до рахункового множинаі кінцевих уявлень, оскільки потрібно мати можливість записати таке кінцеве представлення у вигляді рядка символів кінцевої довжини. Тому мов значно більше, ніж кінцевих уявлень.

По-третє, можна запитати, яка структура тих класів мов, для яких існує кінцеве уявлення?

Представлення мови.

Процедура - це кінцева послідовність інструкцій, які можуть бути механічно виконані. Прикладом може служити машинна програма. Процедура, яка завжди закінчується, називається алгоритмом.

Один із способів подання мови - дати алгоритм, що визначає, чи належить ланцюжок мови. Більш загальний спосіб полягає в тому, щоб дати процедуру, яка зупиняється з відповіддю «так» для ланцюжків, що належать мові, і або зупиняється з відповіддю «ні», або взагалі не зупиняється для ланцюжків, які не належать мові. Кажуть, що така процедура або алгоритм розпізнає мову.

Такий метод являє мову з точки зору розпізнавання. Мова можна також представити методом породження. А саме, можна дати процедуру, яка систематично породжує в певному порядку ланцюжка мови.

Якщо ми можемо розпізнати ланцюжка мови над алфавітом V або за допомогою процедури, або за допомогою алгоритму, то ми можемо і генерувати мову, оскільки ми можемо систематично генерувати всі

ланцюжки з V *, перевіряти кожну ланцюжок на приналежність мови та видавати список тільки ланцюжків мови. Але якщо процедура не завжди закінчується при перевірці ланцюжка, ми не зрушимо далі першої ланцюжка, на якій процедура не закінчується. Цю проблему можна обійти, організувавши перевірку таким чином, щоб процедура ніколи не продовжувала перевіряти один ланцюжок нескінченно. Для цього введемо наступну конструкцію.

Припустимо, що V має p символів. Ми можемо розглядати ланцюжки з V * як числа, представлені в базисі p, плюс порожній ланцюжок e. Можна занумерувати ланцюжка в порядку зростання довжини і в «числовому» порядку для ланцюжків однакової довжини. Така нумерація для ланцюжків мови {a, b, c} * наведена на рис. 2.1, а.

Нехай P - процедура для перевірки приналежності ланцюжка мови L. Припустимо, що P може бути представлена дискретними кроками, так що має сенс говорити про i-му кроці процедури для будь даного ланцюжка. Перш ніж дати процедуру перерахування ланцюжків мови L, дамо процедуру нумерації пар позитивних чисел.

Всі впорядковані пари позитивних чисел (x, y) можна відобразити на множина позитивних чисел наступною формулою:

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

Пари цілих позитивних чисел можна впорядкувати у відповідності зі значенням z (рис. 2.1, б).

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

а

x \ y	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

Тепер можна дати процедуру перерахування ланцюжків L. Нумеруючи впорядковані пари цілих позитивних чисел - (1,1), (2,1), (1,2), (3,1), (2,2), ... При нумерації пари (i, j) генеруємо i-ю ланцюжок з V * і застосовуємо до ланцюжку першою j кроків процедури P. Як тільки ми визначили, що згенерувала ланцюжок належить L, додаємо ланцюжок до списку елементів L. Якщо ланцюжок і належить L, це буде визначено P за j кроків для деякого кінцевого j. При перерахуванні (i, j) буде згенерована ланцюжок з номером і. Легко бачити, що ця процедура перераховує всі ланцюжки L.

Якщо ми маємо процедуру генерації ланцюжків мови, то ми завжди можемо побудувати процедуру розпізнавання речень мови, але не завжди алгоритм. Для визначення того, чи належить х мові L, просто нумеруючи пропозиції L і порівнюємо х з кожним реченням. Якщо згенеровано х, процедура зупиняється, розпізнавши, що х належить L. Звичайно, якщо х не належить L, процедура ніколи не закінчиться.

Мова, пропозиції якої можуть бути згенеровані процедурою, називається рекурсивно перелічуваною. Мова рекурсивно перерахуємо, якщо мається процедура, що розпізнає пропозиції мови. Кажуть, що мова рекурсівен, якщо існує алгоритм для розпізнавання мови. Клас рекурсивних мов є власним підмножиною класу рекурсивно перелічуваною мов. Мало того, існують мови, які не є навіть рекурсивно перелічуваною.