**Граматики, що породжують (граматики Н. Хомського)**

Граматики, що породжують, - це простий і потужний механізм, що дозволяє задавати великий клас мов, що містять безліч ланцюжків.

За допомогою граматик, що породжують, ми зможемо, зокрема, визначити мови L3, L4 і L5, для завдання яких ми раніше обмежувалися словесними формулюваннями. Граматики, що породжують, використовуються і при описі синтаксису мов програмування.

Граматикою, що породжує, називається четвірка:

**G = (T,N,P,S),**

де Т - кінцева множина термінальних (основних) символів - основний алфавіт. Елементами множини Т є символи, з яких зрештою і складаються ланцюжки мови, що породжується даною граматикою. Термінальний (від латів. terminus - межа, кінець) і означає "кінцевий, ".

Т — це не що інше, як алфавіт мови, що породжується граматикою. Надалі, якщо не зазначено особливо, термінальні символи, чи навіть термінали, позначатимуться малими літерами латинського алфавіту: а,b,с тощо.;

N - кінцева множина нетермінальних (допоміжних) символів - допоміжний алфавіт. Нетермінальні символи, інакше нетермінали, - це поняття граматики (мови), які використовуються при його описі. Нетермінали позначатимемо великими латинськими літерами: А, В, C, D, E і т. д.;

Р — кінцева множина правил виведення, званих також продукціями. Кожне правило множини Р має вигляд:

**α** → **β**,

де **α** і **β** - ланцюжки термінальних та нетермінальних символів. Ланцюжок **α** не повинен бути порожнім, ланцюжок **β** може бути порожнім:

**α** ∈ (**Т** ∪ **N**)+;

**β** ∈ (**Т** ∪ **N**)\*

Правило **α** → **β** визначає можливість підстановки **β** замість **α** у процесі виведення (породження) ланцюжків мови;

**S** (S ∈ N) — початковий символ граматики — один із множини нетермінальних символів, початковий (стартовий) нетермінал. Початковий нетермінал - це поняття, що відповідає правильній пропозиції мови. Наприклад, початковий нетермінал граматики виразів означає "вираз", а початковий нетермінал граматики мови Паскаль - "програма".

**Приклад граматик. Породження речень мови**

**Приклад 1**. Розглянемо граматику

G1 = ( {а, b}, {S}, {S → aSb, S → **ε** }, S).

Тут усі елементи четвірки записані явно. Множина термінальних символів Т = {а, b}; множина нетерміналів містить один елемент: N = {S}, а множина правил — два: Р = {S →aSb, S → ε }; роль початкового нетерміналу виконує S.

Граматика може використовуватися для породження (виведення) ланцюжків - речень мови. Процес породження починається з початкового нетерміналу, у прикладі це S. Якщо серед правил є таке, у лівій частині якого записан ланцюжок S, то початковий нетермінал може бути замінений правою частиною будь-якого з таких правил. Обидва правила граматики G1 містять у лівій частині S.

Застосуємо підстановку, задану першим правилом, замінивши S на aSb:

S aSb.

До ланцюжка, що вийшов aSb знову, якщо вдасться, можна застосувати одне з правил граматики. Якщо в ланцюжку є підчіплення, що збігається з лівою частиною хоча б одного з правил, то цей підчіпок можна замінити правою частиною будь-якого з таких правил. У ланцюжку aSb є підчіпок S, що збігається з лівою частиною обох правил граматики G1. Ми маємо право застосувати будь-яке з цих правил.

Використовуємо знову правило (1) для продовження виведення:

SaSbaaSbb.

Тепер до ланцюжка, що вийшов, застосуємо правило (2) {S → ε), замінивши S порожнім ланцюжком. Отримаємо таку послідовність підстановок (саму букву ε в останньому ланцюжку записувати, звичайно, не потрібно):

S aSb aaSbb aabb.

Очевидно, що до ланцюжку, що вийшов, жодне з правил граматики G1 більше не застосовується. Процес породження завершено.

Неважко помітити, що за допомогою граматики G1 можна породити будь-який ланцюжок мови L1 = {anbn | n ≥ 0}, застосувавши до початкового нетермінала правило (1) (S → aSb) n разів, а потім один раз правило (2).

У той же час граматика G1 не породжує жодного ланцюжка термінальних символів, що не належить мові L1. Це множина термінальних ланцюжків, що породжуються граматикою G1 збігається з мовою L1. Іншими словами, граматика G1 породжує мову L1.

L(G1)=L1.

Зазвичай під час запису граматики не виписують четвірку її елементів явно. При дотриманні угод про позначення терміналів та нетерміналів достатньо записати лише правила. Правила з однаковою лівою частиною можна поєднувати, відокремлюючи альтернативні праві частини вертикальною межею. Першим записується правило, що містить у лівій частині початковий нетермінал. З огляду на це граматика G1 може бути записана так:

G1: S →aSb | **ε** .

**Приклад 2**. Розглянемо граматику G2 (цифри справа – номери правил)

G2: S→aSBc (1)

S→abc (2)

сВ → Bc (3)

bВ → bb (4)

S→ **ε** (5)

Проведемо виведення ланцюжків із початкового нетерміналу граматики G2. Під стрілкою, що означає підстановку, будемо вказувати, як і раніше, номер застосованого правила. Отже:

SaSBcaabcBcaabBcc ааbbсс.

Ще одна серія підстановок:

SaSBcaaSBcBcaaabcBcBc aaabBccBcaaabBcBcc

aaabBBcccaaabbBccc аааbbbссс.

Можна переконатися, що граматика G2 породжує ланцюжки терміналів виду

anbncn

і жодні інші. Кількість повторень символів у результуючій ланцюжку визначається тим, на якому кроці застосовується правило (2). Наявність правила (5) дозволяє отримати порожній ланцюжок, якщо застосувати це правило першим, тоді як спроба використання цього правила на наступних кроках не дозволить вивести ланцюжок терміналів. Граматика G2 породжує множину термінальних ланцюжків, що збігається з мовою L2 = {anbncn| n ≥ 0}:

L(G2) = L2.

**Приклад 3**. Граматика, що породжує мову правильних виразів язика (мова Діка).

G3: S→(S) (1)

S→SS (2)

S→ **ε** (3)

Неважко зрозуміти логіку побудови правил цієї граматики. Сенс першого правила такий: уклавши в дужки правильний скобковий вираз S, ми знову отримаємо правильний скобковий вираз. Друге правило означає, що дві правильні дужки, записані одне за одним, дають новий правильний вираз. Нарешті, за правилом (3) порожній ланцюжок вважається правильним виразом. Якби ми вирішили, що не слід дозволяти порожні вирази, правило (3) можна замінити на S → ().

Приклад 4. Граматика найпростіших арифметичних виразів. Єдиним нетерміналом цієї граматики (він же початковий) буде *Вираз*:

G4:

*Вираз*→*Вираз*+*Вираз* |

*Вираз* - *Вираз* |

*Вираз*\* *Вираз* |

*Вираз* / *Вираз*

| а |b |с | ( Вираз ) .

Така граматика породжує ланцюжки терміналів, які є правильними арифметичними виразами. Символи а,b і с в таких виразах позначають операнди, а «+», «-», «\*», і «/» — знаки операцій. Дозволяються круглі дужки (у тому числі вкладені). Приклади правильних виразів:

(a+b)/(b\*c),

(а).

Всі операції двомісні, унарні плюс і мінус не передбачені, тому, наприклад, ланцюжок -а не належить мові, що породжується цією граматикою.

Вираз – одне з основних понять мов програмування. Надалі граматиці виразів буде приділена чимала увага. Щоб запис цих граматик був коротшим, замінимо нетермінал Вираз на Е (від expression - вираз), повернувшись тим самим до прийнятих раніше угод про іменування нетерміналів. Тоді граматика, яку позначимо G5, запишеться так:

G5: E→E + E|E-E|E\*E|E/E|a|b|c| (E).

Вочевидь, що вона породжує таку ж мову, як і граматика G4. А саме:

L(G5) = L5.