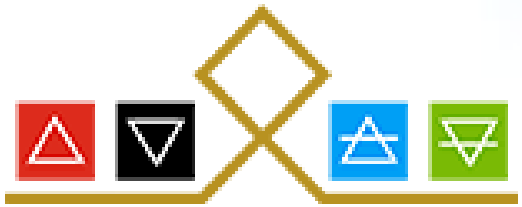
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ



Теорія розпізнавання образів та класифікації в системах штучного інтелекту

Тема №6. Елементи теорії трансляції

Київ - 2025

Зміст

Алфавіт — скінченна множина символів. ε — порожній ланцюжок, слово, послідовність. Алфавітом є об'єднання алфавітів, перетин, різниця алфавітів.

Нехай T — алфавіт, тоді:

- T^+ — множина усіх можливих послідовностей, що складені з елементів цього алфавіту крім порожньої послідовності ε.
- T* — множина усіх можливих послідовностей, що складені з елементів цього алфавіту, будь-якої довжини. Отже: $* = + \cup \{\ \}$
- T^k — множина усіх можливих послідовностей,що складені з елементів цього алфавіту, довжини не більше k.

Мова в алфавіті T це множина ланцюжків скінченної довжини в цьому алфавіті. Кожна мова в алфавіті T є підмножиною множини T*.

Формальна граматика - це четвірка G ={T,N,P,S}. Де:

* **T** — алфавіт термінальних символів, терміналів (від англ. terminate - завершитись). Термінальні символи є алфавітом мови.
* **N** — алфавіт нетермінальних символів, нетерміналів. T∩N =∅; Нетермінали не входять в алфавіт мови.
* **S** — аксіома, спеціально виділений нетермінальний символ з якого починається опис граматики.
* S∈N**P** — правила виводу, скінченна підмножина множини (∪)+×(∪)∗.

Інколи **P** визначають так: ∈(∪)∗× ×(∪)∗, ∈(∪)∗. Елемент (α,β) з множини P називається *правилом виводу* і записується у вигляді → . Таким чином, ліва частина правила не може бути порожньою. Правила з однаковою лівою частиною записують: → 1| 2|...| , , i = 1,2,..,n - називаються альтернативами правила виводу з ланцюжка .

Тексти будь-якою мовою (природною чи формальною) є ланцюжками символів деякого алфавіту.

Ланцюжком символів називають довільну впорядковану кінцеву послідовність символів, записаних один за одним.

Концепція **символу** є базовою для теоретично формальних мов. Для ланцюжка символів мають значення три фактори: склад символів, що входять у ланцюжок, їх кількість і порядок символів у ланцюжку. Далі ланцюжки символів позначатимемо грецькими літерами: α, β, γ та ін.

Ланцюжки символів α та β **рівні**(α = β), якщо вони мають один і той самий склад символів, те саме їх кількість і однаковий порядок слідування символів у ланцюжку.

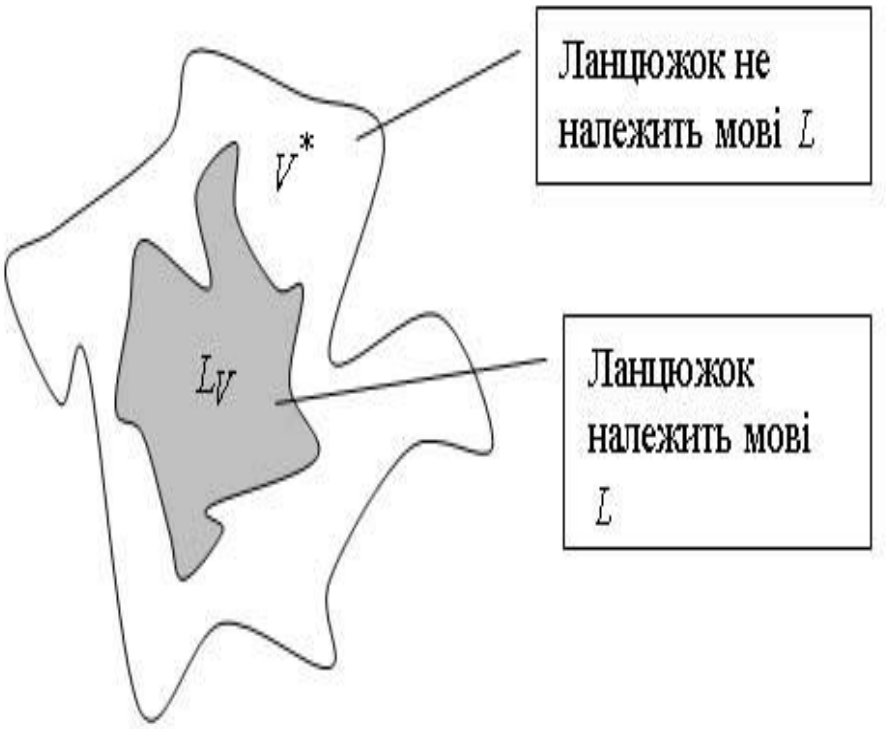
Кількість символів у ланцюжку визначає його **довжину**. Довжина ланцюжка α позначається як | α |.

1. Означення формальних мов, ланцюжки

Позначимо $V^+ = V^* \setminus \{e\}$ – множину всіх слів, крім е (ε).

Припустимо, що ми маємо слово ω, тоді послідовність

$\underbrace{\omega\omega\omega\ldots\omega}_{n} = \omega^n$ а $\omega^0 = \varepsilon$



Формальне означення ланцюжка:

1) ε –ланцюжок в V.

2) Якщо – ω ланцюжок в алфавіті V і а є V, то – ωа ланцюжок в V.

3) α– ланцюжок в алфавіті V, якщо він ланцюжок внаслідок 1) або 2).

Приклади формальних мов

1. Множина всіх слів в алфавіті $V_1 = \{a\}$, $L_1 = \{\varepsilon, a, aa, aaa, ...\} = \{a^n \mid n \geq 0\}$

2. Нехай $V_2 = \{a, b, 1\}$, L_2 визначає множину ідентифікаторів:

$$L_2 = \{a, b, a1, b1, aa, ab, aa1, ...\}.$$

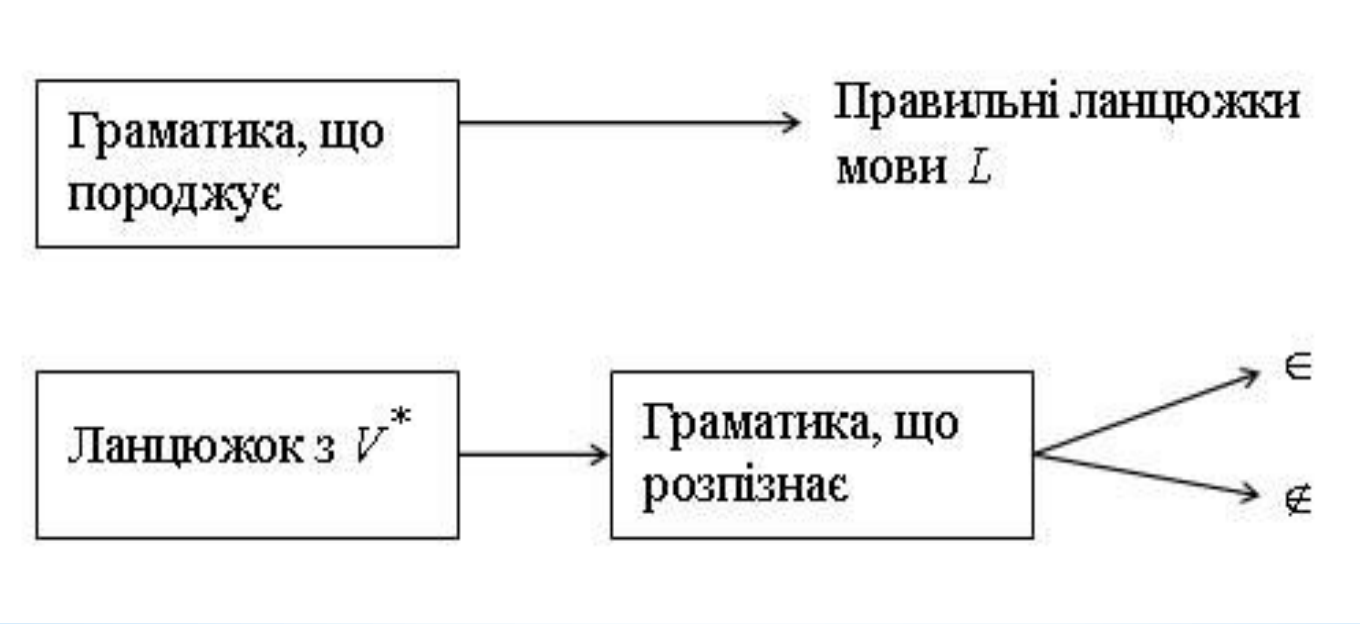
3. $V_3 = \{a, b, c\}$, $L_3 = \{a^n bc^n \mid n \geq 0\}$, $aabcc \in L_3$, $abcc \notin L_3$.

4. $V_4 = \{a\}$, $L_4 = \{\alpha \in V_4 \mid |\alpha| = 2^k, k = 0,1,2,...\}$, $aa \in L_4$, $aaaa \in L_4$, $aaa \notin L_4$.

5. V_5 = {a, b, c}; L_5 = {$a^n bc^m$ | n, m ≥ 0}, aaaabcc ∈ L_5, cbaa ∉ L_5.

6. V_6 = {a, b, c}; L_6= {x | у ланцюжку х кількості входжень a, b і c рівні}, cbaba ∈ L_6, ccb ∉ L_6.

Задача належності, способи визначення мов



Регулярні операції над мовами

Нехай є три мови: L_1, L_2, L з алфавіту V.

Об'єднання $L_1 \cup L_2$ буде визначати нову мову:

$$L_1 \cup L_2 = \{\omega \mid \omega \in L_1 \text{ або } \omega \in L_2\}$$

Приклад: $\{a, ab, ba\} \cup \{a, ba, bb\} = \{a, ab, ba, bb\}$.

Конкатенацією (катенацією) мов L_1 **і** L_2 називається

$$L_1 L_2 = \{\omega_1 \omega_2 \mid \omega_1 \in L_1, \omega_2 \in L_2\}$$

Приклад: $\{a, b\}\{c, d, e\} = \{ac, ad, a, bc, bd, b\}$.

Піднесення до степеня мови L. $L^0 = \{e\}$. $L^i = L^{i-1} L$.

Приклад: $\{e, a, aa\}^2 = \{e, a, aa\}\{e, a, aa\} = \{e, a, aa, aaa, aaaa\}$.

Ітерація мови: $L^* = \{\omega^i \mid \omega \in L, i \geq 0\} = \{e\} \cup L \cup L^2 \cup \ldots \cup L^n \cup \ldots$.

2. Метамова БНФ (Бекуса-Наауера)

У кожній мові є своя *система понять* (будь-який конкретний оператор є представником загального поняття "оператор", будь-яке ім'я – представником поняття "ім'я" тощо). Представники понять (конкретні оператори або імена – це вирази деякої структури (синтаксису). Наприклад, усі імена – це послідовності букв і цифр, що починаються з букви, цілі сталі – послідовності цифр, а кожний оператор присвоювання складається з імені, знака ":=" і виразу. Остання фраза по суті містить три правила: вони описують синтаксис представників понять "ім'я", "стала", "оператор присвоювання" і називаються *синтаксичними*.

Синтаксичні правила.

Позначимо поняття словами в <кутових дужках>. Це позначення розглядається як неподільне і називається *нетермінальним символом*, або *нетерміналом*, наприклад, <оператор> або <ім'я>. Символи й лексеми мови будемо брати в 'апострофи' або виділяти жирним шрифтом, наприклад, program або ':='. Вони також розглядаються як неподільні і називаються *термінальними символами*, або *терміналами*.

«Термінальний" означає "остаточний", тобто термінали – це і є "остаточні" символи мови.

"Нетермінальний» ("неостаточний«) символ не є символом мови (позначення представників якогось поняття, а їх структура повинна бути описана синтаксичними правилами). Наприклад, вигляд терміналів '+', ':=' або program зафіксовано в мові Паскаль, а структуру представників понять <оператор присвоювання> або <ім'я> треба описати.

Послідовність, складена з терміналів і нетерміналів, називається *метавиразом*, наприклад, <ім'я> ':=' <вираз>. Елементи метавиразу, тобто нермінальні й нетермінальні символи (іноді відокремлюються пропусками). Порожня послідовність позначається кутовими дужками <>.

Фраза "оператор присвоювання складається з імені, знака ":=" і виразу« :

<оператор присвоювання> *має структуру* <ім'я> ':=' <вираз>.

Замість слів "*має структуру*« знак "::=" і одержимо:

<оператор присвоювання> ::= <ім'я> ':=' <вираз>.

Взагалі, усяку фразу вигляду

<поняття> *має структуру* <метавираз> можна переписати у вигляді:

<поняття> ::= <метавираз>.

Синтаксичні правила, записані у вигляді <поняття> ::= <метавираз>, називаються *формами Бекуса-Наура*, за прізвищами тих, хто їх придумав. Форми Бекуса-Наура скорочено називаються БНФ. Поняття, записане в БНФ ліворуч від "::=", називається її *лівою частиною*, а метавираз праворуч – *правою*. Знак "::=" не є символом мови й називається *метасимволом*.

Сама по собі БНФ

<оператор присвоювання> ::= <ім'я> ':=' <вираз>

задає лише загальну структуру кожного з представників поняття "оператор присвоювання", але не їх конкретний вигляд. Для цього треба описати структуру представників понять <ім'я> і <вираз>. Пригадаємо: "ім'я – це послідовність букв і цифр, що починається з букви". У цій фразі виникають одразу два нові поняття – <буква> і <послідовність букв і цифр>. Перепишемо її у вигляді БНФ

<ім'я>::=<буква><послідовність букв і цифр>.

На цьому поки що зупинимося. Очевидно, для описання синтаксису останніх двох понять потрібні будуть свої БНФ, можливо, з новими поняттями. У всякому разі, зараз ми припустимо, що

синтаксис виразів мови задається деякою сукупністю БНФ, або синтаксичних правил.

Сукупність БНФ задає синтаксис виразів мови.

Якщо структура представників якогось поняття задається кількома БНФ, то їх об’єднують, записавши альтернативні праві частини в однім правилі й відокремивши символом "|". Цей символ позначає слово "або"; він також є метасимволом.

З цими позначеннями очевидні такі БНФ:

<означення> ::= великий | злющий

<підмет> ::= комар | слон

Підмет у реченні може бути як із означенням, так і без нього.

Поняття <група підмета> і БНФ

<група підмета> ::= <означення> <підмет> | <підмет>

Тоді структура речення задається такою БНФ:

<речення> ::= <група підмета> <присудок>;

Серед понять мови виділяється *головне*; воно позначається спеціальним *початковим нетерміналом*. Очевидно, що в нашій мові, наприклад, головним поняттям є *речення*, а в мові Паскаль – *програма*.

Поняття *послідовність терміналів - вивідна з початкового нетермінала* (*формальна мова*, задана сукупністю БНФ).

Якщо замінити початковий нетермінал (позначимо його *S*) на праву частину правила, у якому *S* ліворуч, то одержимо послідовність символів (терміналів і нетерміналів), що називається *вивідною з S*. У прикладі 10.1 такою є

<група підмета> <присудок>

Якщо у вивідної з *S* послідовності замінити якийсь нетермінал на відповідну йому праву частину, то одержимо послідовність, що теж називається вивідною з *S*, тощо.

Вивідні з *S* послідовності, що складаються лише з терміналів, називаються *вивідними виразами (* представниками головного поняття мови. Наприклад, послідовність злющий комар тупотить є вивідним виразом і представником головного поняття – речення.

Формальна мова, задана сукупністю БНФ – це множина вивідних виразів.

Крім поняття виводимості з початкового нетермінала, використовується також поняття виводимості з довільної послідовності терміналів і нетерміналів незалежно від того, чи виводиться сама ця послідовність із *S*, чи ні.

Будь-яка з альтернатив метавиразу виводиться з нього.

Наприклад, із метавиразу

<група підмета> ::= <означення> <підмет> | <підмет>

виводяться і <означення> <підмет>, і <підмет>.

Розглянемо оператори присвоювання змінним, іменами яких можуть бути лише x, y, z, а вирази у правій частині можуть бути або сталими 1 і 2, або іменами x, y, z, або сумою чи різницею цих сталих і змінних. Головним тут, очевидно, є поняття <оператор присвоювання>:

<оператор присвоювання> ::= <ім'я> ':=' <вираз>

Вираз складається зі сталих і імен. Узагальнимо їх поняттям <первинне>, і запишемо БНФ виразів і первинних:

<вираз> ::= <первинне> | <первинне> '+' <первинне> |

<первинне> '-' <первинне>

<первинне> ::= <стала> | <ім'я>

БНФ сталих і імен очевидні:

<стала> ::= '1' | '2'

<ім'я> ::= 'x' | 'y' | 'z‘

насправді є окремим випадком формальної метамови – *мови формальних граматик*.

Записана сукупність БНФ задає синтаксис операторів присвоювання, а також виразів, сталих і імен. Крім того, задано множини конкретних імен, сталих, виразів і операторів присвоювання.

БНФ – це вираз у алфавіті, що складається з терміналів, нетерміналів і спеціальних метасимволів. БНФ мають цілком визначений синтаксис (нетермінал, потім знак '::=' і метавираз). Їхньою семантикою є задання структури і множин представників понять, позначених нетерміналами. Таким чином, ми маємо *мову БНФ*. Вона призначена для описання інших мов і називається *метамовою*.

Існують різні метамови; деякі з них задаються строго й точно засобами логіки і математики і тому називаються *формальними*. Мова БНФ, описана тут неформально, насправді є окремим випадком формальної метамови – *мови формальних граматик*.

Еквівалентність БНФ

Дві сукупності БНФ називаються *еквівалентними*, якщо задають ту саму формальну мову.

2. Метамова БНФ

<поняття> *має структуру* **<метавираз>**

<поняття> ::= <метавираз>

БНФ оператора присвоєння:

<оператор присвоювання> ::= <ім'я> ':=' <вираз>

<ім'я>::=<буква><послідовність букв і цифр >

Приклад 1. <оператор присвоювання> ::= <ім'я> ':=' <вираз>

<вираз>::=<первинне>|<первинне>'+’<первинне>|<первинне>'-’<первинне>

<первинне<стала>|<ім'я>

<стала>::= '1' | '2'

<ім'я>::= 'x' | 'y' | 'z'

3. Розширені БНФ

Означення: Метавирази з символами "(", ")", "[", "]", "{", "}" нази-ваються ***розширеними***, а БНФ – ***розширеними БНФ***, або ***РБНФ***.

$X, Y, Z, \ldots, T$ – довільні метавирази (можливо, порожні), N – нетермінал

$$N ::= X\ Z\ Y$$

$$\cdots$$

$$N ::= X\ T\ Y$$

$$N ::= X\ (\ Z \mid \ldots \mid T\ )\ Y$$

<вираз> ::=
<первинне> '+' <первинне> |
<первинне> '-' <первинне>

<вираз> ::=
<первинне> ('+' | '-') <первинне>

$N ::= X\ Z\ Y$ $N ::= X\ Y$	$N ::= X\ [\ Z\ ]\ Y$
<вираз> ::= <первинне> | <первинне> ('+'| '-') <первинне>	<вираз> ::= <первинне> [('+'| '-') <первинне>]
<оператор присвоювання> ::= <ім'я> ':=' <вираз> <вираз> ::= <первинне> | <первинне> '+' <первинне> | <первинне> '-' <первинне> <первинне> ::= <стала> | <ім'я> <стала> ::= '1' | '2' <ім'я> ::= 'x' | 'y' | 'z'	<оператор присвоювання> ::= <ім'я> ':=' ('1' | '2' | <ім'я>) [('+'| '-') ('1' | '2' | <ім'я>)] <ім'я> ::= 'x' | 'y' | 'z'

Ітераційні дужки “{“ , “}”

Якщо *X* – довільний метавираз, то *метавираз {X} позначає всі послідовності (у тому числі порожню) виразів, вивідних із X.*

Прикляд. Визначимо записати поняття «ідентифікатор» в алфавіті V={A,B,C,0,1}

<Ід> ::=<Б> { <Б> | <Ц> } <Б> ::= 'A' | 'B' | 'C' <Ц> ::= '0' | '1'	<Ід> ::= ('A' | 'B' | 'C'){ 'A' | 'B' | 'C' | '0' | '1' }

4. Граматики Хомського, основні поняття

Позначимемо формальну граматику $G = (N, \Sigma, P, S)$.
N - множина нетермінальних символів;
Σ - множина термінальних символів;
P - множина правил виводу;
S - початковий символ.

Порядок застосування правил довільний. Граматика визначає лише правила, які дозволено використовувати у поточній ситуації, і не дає вказівки, які правила мають бути застосовані.

Формальні граматики дозволяють описувати складніші формальні мови. Наприклад, синтаксис мови програмування Паскаль визначено у вигляді формальної граматики.[1]

1. *Сума* → *Цифра*
2. *Сума* → *Сума* + *Цифра*
3. *Сума* → *Сума* - *Цифра*
4. *Цифра* → 1..9

До першого речення можна застосувати правила 1—3, та не можна застосувати 4 правило. Починаючи з 4 речення можливе застосування лише 4 правила.

Аксіомою в цій мові є *Сума*. Виходячи з неї, застосуванням наведених вище правил, можна отримати необхідний вираз:

Сума	Аксіома
Сума – *Цифра*	Застосовано 3 правило
Сума + *Цифра* – *Цифра*	Застосовано 2 правило
Цифра + *Цифра* – *Цифра*	Застосовано 1 правило
1 + *Цифра* – *Цифра*	Застосовано 4 правило
1 + 2 – *Цифра*	Застосовано 4 правило
1 + 2 – 9	Застосовано 4 правило

Порядок застосування правил довільний. Граматика визначає лише правила, які дозволено використовувати у поточній ситуації, і не дає вказівки, які правила мають бути застосовані.

Формальні граматики дозволяють описувати складніші формальні мови

Наприклад, синтаксис мови програмування Паскаль визначено у вигляді формальної граматики.

Ієрархія Хомського намагається класифікувати необмежені, в принципі, мови, символьні вирази та правила. Для цього запроваджують класи мов, для яких можливо встановити швидкодію та підхід до комп'ютерної обробки.

Мови тим більше обмежені, чим глибше знаходяться в ієрархії. Взагалі, можна помітити, що простіші мови, з одного боку, простіше піддаються комп'ютерній обробці, а з іншого — їм бракує виразності.

Наприклад, для пошуку деяких шаблонів у тексті використовують регулярні вирази.

Регулярні вирази відповідають граматикам Хомськогоі типу 3. Просто їхньої виразності досить, аби шукати різноманітні фрагменти тексту, але їх не достатньо, наприклад, описання мов програмування. Для описання мов програмування використовують, зазвичай, граматики типу 2.

Чотири типи формальних граматик, правил виводу, формальних мов та автомати, здатні їх розпізнати

Граматика	Правила	Мови	Автомати	Скорочення
Тип-0 Довільна формальна граматика	$\alpha \to \beta$ $\alpha \in V^* N V^*, \beta \in V^*$	рекурсивно зліченна	Машина Тюринга	KSV*
Тип-1 Контекстно-залежна граматика	$\alpha A \beta \to \alpha\gamma\beta$ $A \in N, \alpha, \beta \in V^*, \gamma \in V^+$ $S \to \varepsilon$ дозволене, коли серед правил P відсутнє $\alpha \to \beta S\gamma$.	контекстно-залежна	Лінійний обмежений автомат	КЗ
Тип-2 Контекстно-вільна граматика	$A \to \gamma$ $A \in N, \gamma \in V^*$	контекстно-вільна	недетермінований автомат з магазинною пам'ятю	КВ
Тип-3 регулярна граматика	$S \to \varepsilon$ $A \to aB$ (праволінійна) або $A \to Ba$ (ліволінійна) $A \to a$ $A \to \varepsilon$ $A, B \in N, a \in \Sigma$	регулярна	Скінченний автомат (як детермінований, так і недетермінований)	А

Для описання мов програмування використовують, зазвичай, граматики типу 2.

4. Граматики Хомського, основні поняття

Означення: Граматикою Хомського називається четвірка

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

N – множина ***позначень понять*** мови, тобто ***нетермінальних символів*** (***нетерміналів***).

T – алфавіт означуваної мови, або множина ***термінальних символів*** (***терміналів***). $T \cap N = \varnothing$

P – множина ***правил виведення*** (***продукцій***) вигляду $v \to w$, де

$$v \in (T \cup N)^* N (T \cup N)^*, \; w \in (T \cup N)^*$$

тобто правий ланцюжок є довільною послідовністю терміналів і нетерміналів, а лівий містить принаймні один нетермінал.

S – ***початковий нетермінал*** із множини N, або ***позначення головного поняття***, яким позначаються слова мови.

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

Можна вважати P – скінченною підмножиною такої множини:

$$(T \text{È} N)^* N (T \text{È} N)^* \times (T \text{È} N)^*$$

Якщо (ν,ω) належить множині P, то серед правил граматики G існує правило вигляду $\nu \rightarrow \omega$.

Приклад:

$(AB, CDE) \in P$ $\qquad$ $AB \rightarrow CDE$

Якщо $\alpha = xyABLC$, $\alpha \in G$, то $\beta = xyCDELC \in G$.

ν® w_1 і ν® w_2	ν® $w_1 \mid w_2$
ν® $w_1 w w_2$ і ν® $w_1 w_2$	ν® $w_1[w]w_2$
ν® $w_1 u_1 w_2$ і ν® $w_1 u_2 w_2$	$\nu 1$® $w1(u1 \mid u2)w2$

Приклади граматик Хомського

Приклад . $G_0 = (N, T, P, S)$

N: $\{ A, S \}$

T: $\{0,1\}$

P: $S \to 0A1,\ 0A \to 00A1,\ A \to e$

Приклад . $G_1 = (\{ A, B, C, D \}, \{ a, 1, 2 \},$

$\{ A \to BC,\ A \to BD,\ A \to B,\ B \to a,\ C \to 1,\ D \to 2 \},\ A)$

P можна переписати у вигляді

$\{ A \to B [C \mid D],\ B \to a,\ C \to 1,\ D \to 2 \}$

Поняття

1. На множині слів об'єднаного алфавіту $(TÈ\ N)^*$ означається ***відношення безпосередньої виводимості***, позначене знаком $Þ_G$ (або Þ , коли відомо, якою саме є G):

$v Þ_G w$, якщо $v = x_1 u x_2$, $w = x_1 y x_2$, $u ® y Î P$.

При цьому кажуть, що w *безпосередньо виводиться з* v ***застосуванням продукції*** $u ® y$.

Приклад . $G_1 = (\{ A, B, C, D \}, \{ a, 1, 2 \},$

$\{ A ® BC, A ® BD, A ® B, B ® a, C ® 1, D ® 2 \}, A)$

$BCÞ aC$ застосуванням продукції $B® a$,

$aCÞ a1$ застосуванням $C® 1$

2. На множині $(T\grave{E}N)^*$ означається ***відношення виводимості***, позначене $Þ^*_G$ (або $Þ^*$, коли відомо, якою саме є G): v ***Þ* w***, якщо $v=w$ або існує послідовність $w_1, w_2, \ldots, w_n$ слів, де $n^3 1$, така, що $vÞ w_1$, $w_1Þ w_2$, … , $w_{n-1}Þ w_n$, $w_n=w$. Послідовність $vÞw_1Þw_2Þ\ldots Þw_n$ називається ***виведенням*** w_n із v, а n – ***довжиною виведення***.

v Þ* w можна уточнити $v Þ^n w$.

Приклад . $G_1 =(\{ A, B, C, D \},\{ a, 1, 2 \},$

$\{ A ® BC, A ® BD, A ® B, B ® a, C ® 1, D ® 2 \}, A)$

$BCÞ^* a1$, оскільки $BCÞ aC$, $aCÞ a1$. $(BC Þ^2 a1)$

3. Якщо $SÞ_G^* w$, то послідовність $SÞ \ldots Þ w$ називається ***виведенням слова w у граматиці G***, а слово w – ***вивідним***.

Приклад .Слова $A, BC, aC, a1$ вивідні в граматиці G_1.

4. Вивідні слова в алфавіті T називаються ***породжуваними***, а множина їх усіх – ***мовою, що задається*** (***породжується***) ***граматикою G***:

$L(G) = \{w \mid w \hat{I} T^*$ та $S Þ^* w\}$.

Приклад . $G_0 = (N, T, P, S)$

P: $S \to 0A1,\ 0A \to 00A1,\ A \to e$

N: $\{A, S\}$

T: $\{0,1\}$

$L(G_0) = \{0^n 1^n \mid n \geq 1\}$

$$S \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 0011$$

Приклад . $G_1 = (\{A, B, C, D\}, \{a, 1, 2\},$

$\{A \to BC, A \to BD, A \to B, B \to a, C \to 1, D \to 2\}, A)$

$L(G_1) = \{a, a1, a2\}$

$A \to BCÞ aCÞ a1$

5. Вивідний ланцюжок граматики , що не містить нетермінальних символів називається ***термінальним ланцюжком***, що породжується граматикою , а мова, що породжується граматикою – це множина всіх термінальних ланцюжків, що породжуються граматикою.

6. Граматики називаються ***еквівалентними***, якщо задають ту саму мову.

Приклад . $G_1 = (\{ A, B, C, D \}, \{ a, 1, 2 \},$

$\{ A \to BC, A \to BD, A \to B, B \to a, C \to 1, D \to 2 \}, A)$

$\underline{G_1} = (\{ A \}, \{ a, 1, 2 \}, \{ A \to a [1 \mid 2] \}, A)$ $\quad L(G_1) = \{a, a\,1, a\,2 \}$

Приклад . $G_2 = (\{ I, L, D \}, \{ a, \ldots, z, 0, \ldots, 9 \},$

$\{ I \to L \mid IL \mid ID, L \to a \mid \ldots \mid z, D \to 0 \mid \ldots \mid 9 \}, I)$

$\underline{G_2} = (\{I, C\}, \{a, \ldots, z, 0, \ldots, 9\},$

$\{I \to (a|\ldots|z)C, C \to e \mid C(a| \ldots|z|0|\ldots|9)\}, I)$

Позначення

1. Будемо позначати a,b, …,0,1, …9 – термінали.
2. A, B, C, D, E, S – нетермінали.

   E, S – початкові нетермінали.
3. U, V, …, Z – або термінал, або нетермінал.
4. α, β – ланцюжки, що містять термінали і нетермінали.
5. u, v, …,z – ланцюжки, що містять тільки термінали.

5. Класифікація граматик Хомського, приклади

Граматики можна класифікувати за виглядом їх правил. Класифікація нижче називається ієрархією за Хомським.

Нехай $G = (N, T, P, S)$.

Означення 1. Граматика G називається праволінійною (ліволінійною), якщо

$A \to x, A \to xB$, де $A, B \in N, x \in T^*$

(якщо $A \to x, A \to Bx$, де $A, B \in N, x \in T^*$).

Приклад. $G_3 = (\{S\}, \{0,1\}, \{S \to 0S \mid 1S \mid e\}, S)$.

Означення 2. Граматика G називається контекстно-вільною (безконтекстною), якщо кожне правило з P має вигляд:

$$A \to \alpha, A \in N, \alpha \in (T \bigcup N)^* .$$

Застосування продукції *A*® *w* до ланцюжка *uAv* не залежить, тобто є ***вільним*** від сусідніх з *A* символів, які утворюють ***контекст*** *uv*.

Приклад контекстно-вільної граматики:

G4 = ({ E, T,F},{ a, *, +,(,)} ,P,E)

P: E→E+T|T

T→T*F|F

F→(E)|a

E=>E+T=>T+T=>F+T=>a+T=>a+T*F=>a+a*F=>a+a*a

Означення 3. Граматика G називається контекстно-залежною або нескоротною, якщо кожне правило з P має вигляд: $\alpha \to \beta, \alpha, \beta \in (T \cup N)^*, \ |\alpha| \le |\beta|$.

Контекстно-залежні граматики не допускають правила вигляду A →e.

Приклад контекстно-залежної граматики:

G_5 = ({ B,C,S},{ a, b, c} ,P,S)

P:

- S→aSBC | abC
- CB→BC
- bB→bb
- bC→bc
- cC→cc

$$L(G_5) = \{a^n b^n c^n \mid n \ge 1\}.$$

S=>aSBC=>aabCBC=> aabBCC=> aabbCC => aabbcC
=> aabbcc

Означення 4. Граматика, яка не підлягає жодним обмеженням з означень 1,2,3 називається необмеженою або граматикою загального вигляду.

Приклад необмеженої граматики:

$G_6 = (\{ A,B,C,D,S\},\{ a, b\} ,P,S)$

P:

1) $S \to CD$
2) $C \to aCA$
3) $C \to bCB$
4) $AD \to aD$
5) $BD \to bD$
6) $Aa \to aA$
7) $Ab \to bA$
8) $Ba \to aB$
9) $Bb \to bB$
10) $C \to e$
11) $D \to e$

$$S \Rightarrow^{(1)} CD \Rightarrow^{(2)} aCAD \Rightarrow^{(3)} abCBAD \Rightarrow^{(4)} abCBaD \Rightarrow^{(8)} abCaBD \Rightarrow^{(5)}$$
$$\Rightarrow^{(5)} abCabD \Rightarrow^{(10),(11)} abab.$$

$$L(G_6) = (\omega\omega \mid \omega \in \{a,b\}^*).$$

$$L(G_6) = (\omega\omega \mid \omega \in \{a,b\}^*).$$

$$L(G_6) \supseteq (\omega\omega \mid \omega \in \{a,b\}^*).$$

$$L(G_6) \subseteq (\omega\omega \mid \omega \in \{a,b\}^*).$$

1) S→CD
2) C→aCA
3) C→bCB
4) AD→aD
5) BD→bD
6) Aa→aA
7) Ab→bA
8) Ba→aB
9) Bb→bB
10) C→e
11) D→e

$S \Rightarrow CD$

Для $n \geq 1$: $C \Rightarrow^{(2)-(3),n} c_1c_2..c_nCX_nX_{n-1}..X_1$

$\Rightarrow^{(10)} c_1c_2..c_nX_nX_{n-1}..X_1.$

$c_i = a$ тоді і тільки тоді, коли $X_i = A.$

$c_i = b$ тоді і тільки тоді, коли $X_i = B.$

$X_nX_{n-1}..X_1D \Rightarrow^{(4)-(5)} X_nX_{n-1}..X_2c_1D$

$\Rightarrow^{(6)-(9),n-1} c_1X_nX_{n-1}..X_2D$

$\Rightarrow^{*} c_1c_2X_nX_{n-1}..X_3D$

$\Rightarrow^{*(4)-(9)} c_1c_2..c_nD \Rightarrow^{(11)} c_1c_2..c_n.$

$$S \Rightarrow^{*} c_1c_2..c_nc_1c_2..c_n$$

Означення 5. Праволінійна граматика G = (T, N, P, S) називається *регулярною (автоматною)*, якщо

- кожне правило із P, за виключенням S→e, має вигляд A→aB або A→a, де A,B є *N*, a є *T*;
- якщо S→e належить P, то S не зустрічається в правих частинах правил.

Приклади:

G_7 = ({ A,S}, { a, b}, P, S) P: S→abA | e, A→Saa | b

G_8 = ({ A,C,S}, { a, b}, P, S) P: S→ aC | e , A→a, C →bA | bC | e

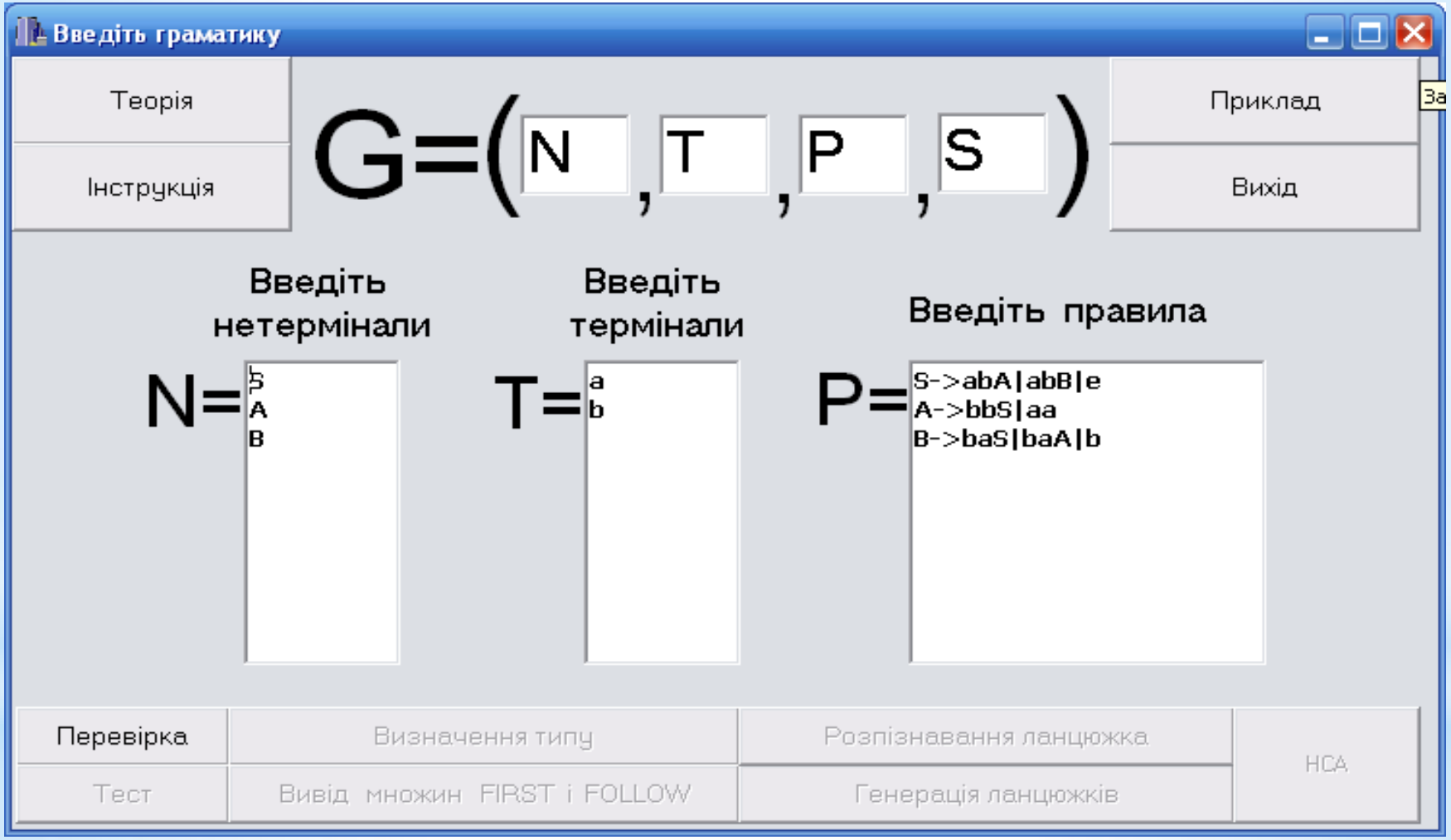
G_9 = ({S}, { 0, 1}, P, S) P: S→0S | 1S | e

Означення 6. Граматика G називається *розширеною граматикою,* якщо вона задається списком пар $A_i \to r_i$, де A_i — різні символи нетермінального алфавіту N; r_i — регулярні вирази в алфавіті $N \cup T$. Нетермінал першої пари вважається головним (початковим).

Приклади розширених граматик:

- $G_{10}=\{S \to AB,\ B \to x|y,\ A \to z|\omega\}$;
- $G_{11}=\{S \to (z|\omega)(x|y)\}$.

$G_{10} \sim G_{11}$

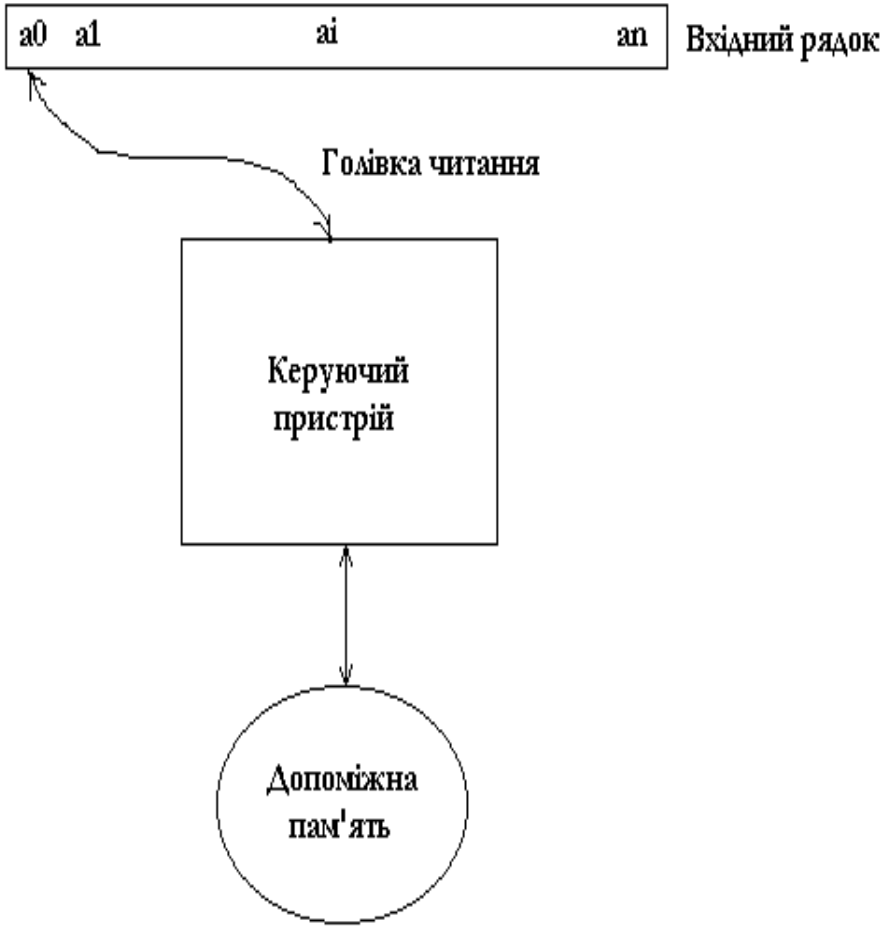

Введіть граматику
Теорія
Інструкція
G=(N , T , P , S)
Приклад
Вихід
Введіть нетермінали
Введіть термінали
Введіть правила
N=
S
A
B
T=
a
b
P=
S->abA|abB|e
A->bbS|aa
B->baS|baA|b
Перевірка
Визначення типу
Розпізнавання ланцюжка
Тест
Вивід множин FIRST і FOLLOW
Генерація ланцюжків
НСА

5. Розпізнавачі

Розпізнавач – це схематизований алгоритм, який визначає деяку множину ланцюжків.

Розпізнавач складається з трьох частин:

- вхідна стрічка;
- керуючий пристрій із скінченню пам'яттю;
- робоча (допоміжна) пам'ять.



Означення 1. Керуючий пристрій називається ***детермінованим***, якщо для кожної конфігурації існує не більше одного наступного детермінованого кроку.

Означення 2. Конфігурація називається ***початковою***, якщо керуючий пристрій знаходиться в заданому початковому стані, вхідна голівка розглядає найлівіший символ, а пам'ять має початкове вмістиме.

Означення 3. Конфігурація називається ***заключною***, якщо керуючий пристрій знаходиться в одному із заключних станів, а вхідна голівка оглядає правий кінець стрічки.

Означення 4. Кажуть, що розпізнавач ***допускає вхідний рядок***, якщо починаючи із початкової кофігурації, розпізнавач може виконати послідовність кроків, що закінчиться заключною конфігурацією.

Означення 5. ***Мова***, що дозволяється розпізнавачем – це множина вхідних ланцюжків, які він допускає.

Для кожного класу граматик із ієрархії Хомського існує свій клас розпізнавачів, які визначають ті самі класи мов, що і граматики.

Приклад:

1) мова L *праволінійна* тоді і тільки тоді, коли вона розпізнається одностороннім детермніованим скінченним розпізнавачем;

2) мова L *контекстно-вільна* тоді і тільки тоді, коли вона визначається одностороннім недетермніованим розпізнавачем з магазинною пам’яттю;

3) мова L *контекстно-залежна* тоді і тільки тоді, коли вона розпізнається двостороннім недетермніованим обмеженим розпізнавачем.