



## Лема про розростання

У теорії формальних мов велике значення мають твердження, в яких формулюється необхідна умова принадлежності мови до того чи іншого класу мов. Ці твердження відомі в літературі за назвою лем про розростання (або лем про “накачування”). За допомогою цих лем вдається довести, що та чи інша мова не є мовою цього класу, наприклад, не є регулярною, не є контекстновільною тощо. Доводити такі “негативні” твердження набагато важче, ніж “позитивні” (що мова є мовою цього класу), бо в останньому випадку достатньо придумати будь-яку граматику відповідного класу, яка породжує цю мову, тоді як в першому потрібно якось довести, що не існує граматики цього класу, яка породжує мову.

Застосування лем про розростання полягає в такому: довівши, що мова не задовольняє умову леми про розростання, ми можемо бути впевнені в тому, що вона не належить до відповідного класу мов.

## Лема про розростання для регулярних мов

У цій лемі стверджується, що будь-яка регулярна мова допускає представлення всіх своїх ланцюжків у вигляді з'єднання трьох ланцюжків, причому середній ланцюжок з цих трьох не є порожнім, обмеженим за довжиною, і його “накачка” – повторення будь-яку кількість разів – або викидання НЕ ВИВОДИТЬ за межі мови (тобто дає ланцюжки, що належать цій регулярній мові).

Якщо  $L$  – регулярна мова, то існує натуральна константа  $k_L$  (залежна від  $L$ ), така, що для будь-якого ланцюжка  $x \in L$ , довжина якого не менша за  $k_L$ ,  $x$  допускає представлення у вигляді  $x = uvw$ , де  $v \neq \lambda$  і  $|v| \leq k_L$ , причому для будь-якого  $n \geq 0$  ланцюжок  $x_n = u v^n w \in L$ .

# Доведення регулярності або нерегулярності мови. Приклад:

Доведемо нерегулярність мови

$$L(M) = \{ a^n b^n, n \geq 0 \}.$$

Вибираючи  $n$  настільки великим, щоб воно перевищувало  $k_L$  (константу леми), одержуємо такі можливі випадки розміщення середнього  $v$  в ланцюжку  $a^n b^n$ . Зокрема можливі варіанти:

$v = a^s$   
 $\underline{a \dots \overbrace{aa \dots a}^n b \dots b}$   
n раз                  n раз

1.  $v = a^s, s < n$ , тобто “накачуваний” підланцюжок  $v$  цілком розташовується в “зоні символів  $a$ ”.

Накачування в цьому випадку виведе за межі мови, оскільки при повторенні ланцюжка  $v$  кількість символів  $a$  необмежено зростатиме, а кількість символів  $b$  залишатиметься сталою.

$v = b^s$   
 $\underline{a \dots a} \overbrace{bb \dots b}^n b \dots b$   
n раз                  n раз

2.  $v = b^s, s < n$ , тобто “накачуваний” підланцюжок  $v$  цілком розташовується в “зоні символів  $b$ ”.

Накачування неможливе з тієї ж причини, що і в попередньому випадку.

$\overbrace{a \dots a}^{a^p} \overbrace{b \dots b}^{b^q} \dots b$   
n раз                  n раз

3.  $v = a^p b^q$ , де  $0 < p < n, 0 < q < n$ , тобто “накачуваний” підланцюжок  $v$  розташовується на стику зон символів  $a$  і  $b$ .

У цьому випадку при накачуванні підланцюжок  $ab$  входить в слово, яке вже не належить мові  $L$ .

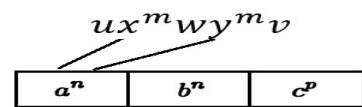
Бачимо, що існують ланцюжки, для яких жодні представлення у вигляді з'єднання трьох ланцюжків не задовільняють умови леми про розростання для регулярних мов.

Отже, мова  $a^n b^n$  нерегулярна.

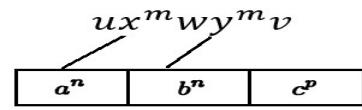
# Лема про розростання для КВ-МОВ.

Для будь-якої КВ-мови  $L$  існує натуральна константа  $k$  (що залежить від  $L$ ), така, що будь-який ланцюжок  $Z \in L$ , довжина якого  $|Z| > 0$ , може бути представлений таким чином у вигляді з'єднання п'яти ланцюжків  $Z = uxwyv$ , де  $|xy| > 0$ ,  $|xwy| \leq k$ , що кожний ланцюжок  $Z_n = ux^nwy^nv$ ,  $n \geq 0$ , належить  $L$ .

Доведення приналежності чи неприналежності мови до класу КВ-мов. Приклад:  
 $L = \{a^n b^n c^p : n, p \geq 0\}$



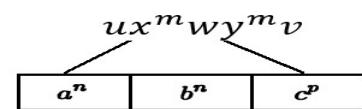
Накачування в цьому випадку виведе за межі мови.



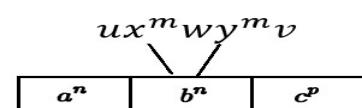
Накачування в цьому випадку не виведе за межі мови.



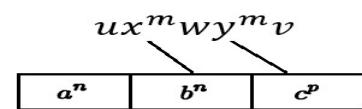
якщо  $|x| = |y|$



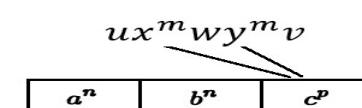
Накачування в цьому випадку виведе за межі мови.



Накачування в цьому випадку виведе за межі мови.



Накачування в цьому випадку виведе за межі мови.

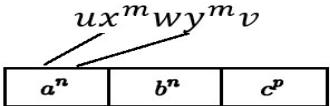


Накачування в цьому випадку не виведе за межі мови.

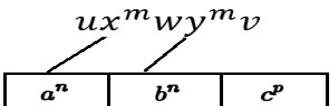


Доведення приналежності чи неприналежності мови до класу КВ-мов. **Приклад №2:**

$$L = \{a^n b^n c^p : n, p \geq 0 \text{ і } p \geq n\}$$

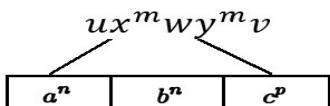


Накачування в цьому випадку виведе за межі мови.

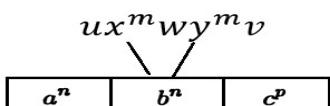


Накачування в цьому випадку виведе за межі мови.

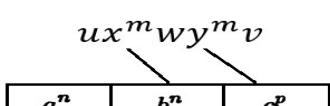
Хоча можна обрати  $|x| = |y|$ , але не завжди виконуватиметься  $p \geq n$ .



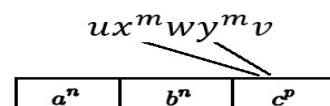
Накачування в цьому випадку виведе за межі мови.



Накачування в цьому випадку виведе за межі мови.



Накачування в цьому випадку виведе за межі мови.



Накачування в цьому випадку виведе за межі мови.

Не завжди виконуватиметься  $p \geq n$ .



# МП-автомати та КВ-граматики.

*Автомат з магазинною пам'яттю* (МП-автомат) – це сімка об'єктів

$$M = (Q, T, \Gamma, D, q_0, Z_0, F), \text{ де}$$

$Q$  – скінчена множина станів керуючого пристрою;

$T$  – скінчений вхідний алфавіт;

$\Gamma$  – скінчений алфавіт магазинних символів;

$D$  – функція переходів, що є відображенням множини  $Q \times (T \cup \{\varepsilon\}) \times \Gamma$  в  $Q \times \Gamma^*$ ,

де

$q_0 \in Q$  – початковий стан керуючого пристрою;

$Z_0 \in \Gamma$  – початковий символ магазина;

$F \subset Q$  – множина заключних станів.

## МП-автомати та КВ-граматики.

Конфігурацією МП-автомата називається трійка  $(q, w, \alpha) \in Q \times T^* \times \Gamma^*$ , де  $q \in Q$  – поточний стан керуючого пристрою;  $w \in T^*$  – непрочитана частина вхідного рядка; перший символ рядка  $w$  знаходиться під головкою читання; якщо  $w = \varepsilon$ , вважають, що весь вхідний рядок прочитаний;  $\alpha \in \Gamma^*$  – вміст магазина; перший зліва символ рядка  $\alpha$  – це верхній символ магазина; у випадку  $\alpha = \varepsilon$ , вважають, що магазин пустий.

## МП-автомати та КВ-граматики.

**Початковою конфігурацією** МП-автомата називається конфігурація вигляду  $(q_0, w, Z_0)$ , де  $w \in T^*$ , тобто керуючий пристрій знаходиться в початковому стані, вхідний рядок містить ланцюжок, який потрібно проаналізувати, а в магазині міститься тільки початковий символ  $Z_0$ .

**Заключна конфігурація** – це конфігурація вигляду  $(q, \varepsilon, u)$ , де  $q \in F$ ,  $u \in \Gamma^*$ , тобто керуючий пристрій знаходиться в одному із заключних станів, а вхідний ланцюжок повністю прочитаний.

За допомогою МП-автоматів можна розпізнавати рядки мови двома способами.

Перший – рядок вважається розпізнаним, якщо після його прочитання автомат переходить у заключний стан, тобто  $(q_0, w, Z_0) \mapsto^* (q, \varepsilon, u)$ , де  $q \in F$ ,  $u \in \Gamma^*$ .

Другий – якщо після прочитання рядка стек пам'яті виявляється порожнім, у цьому випадку говорять, що ланцюжок розпізнається **спустощенням магазина**. Ці два способи еквівалентні.

# МП-автомати та КВ-граматики.

Контекстно-вільна граматика  $G$  це четвірка  $(N, T, P, S)$ :

- $S \in N$
- $N$  та  $T$  скінченні множини, що не перетинаються
- $P$  скінченна підмножина  $N \times (N \cup T)^*$

$N$  — множина нетермінальних символів,

$T$  — множина термінальних символів,

$P$  — множина правил виводу,

$S$  — початковий символ.

Правила  $(\alpha, \beta) \in P$  записують як:

$$\boxed{A \rightarrow \beta}.$$

В лівій частині правила виводу має міститися одна змінна (нетермінальний символ).

Формально, має виконуватись:  $\alpha \in N, \beta \in (N \cup T)^*$ , де  $|\beta| \geq 1$ .

# Низхідний МП-автомата для КВ-граматики. (розглянутий як приклад до контрольного завдання №25) **(Працює як LL(1)-аналізатор)**

Низхідний МП-автомат  $M = (\{q\}, T, N \cup T, D, q, S, q)$ , що розпізнає мову КВ-граматики  $G = (N, T, P, S)$ , моделюється з одним станом  $q$ , вхідним алфавітом є термінальні символи  $T$ -граматики, алфавіт магазинних символів складається з термінальних та нетермінальних символів:  $Z = N \cup T$ . Початкова конфігурація визначається так:  $(q, \alpha, S)$  – автомат перебуває в своєму єдиному стані  $q$ , зчитувальна головка знаходитьться на початку ланцюжка  $\alpha \in T^*$ . У стеку знаходиться початковий символ  $S$ . Кінцева конфігурація автомата визначається так:  $(q, \varepsilon, \varepsilon)$  – автомат перебуває в своєму єдиному стані  $q$ , зчитувальна головка знаходиться під порожнім символом (за кінцем ланцюжка), стек порожній.

# Низхідний МП-автомата для КВ-граматики.

(розглянутий як приклад до контрольного завдання №25)

## (Працює як $LL(1)$ -аналізатор)

Функція переходів МП-автомата будується на основі правил граматики:

- 1)  $(q, \alpha) \in D(q, \varepsilon, A), A \in N, (\alpha \in (N \cup T)^* \text{, якщо } (A \rightarrow \alpha) \in P;$
- 2)  $D(q, a, a) = (q, \varepsilon), \forall a \in T.$

Роботу цього МП-автомата неформально можна описати так:

- якщо в магазині знаходиться нетермінальний символ  $A$ , то його можна замінити на ланцюжок  $\alpha$ , не рухаючи головку зчитування, якщо в граматиці  $G$  є правило  $A \rightarrow \alpha$ ;
- якщо в магазині знаходиться термінальний символ  $a$ , що збігається з поточним символом входного ланцюжка, то цей символ можна викинути з магазина і пересунути головку зчитування на одну позицію праворуч.

# Висхідний МП-автомата для КВ-граматики. (Працює як $LR(0)$ -аналізатор)

Можна побудувати розширений МП-автомат, який працює “знизу догори”, як “висхідний аналіз”, моделюючи в зворотному порядку правосторонні виводи в КВ-граматиці.

*Висхідний МП-автомат*  $M = (\{q\}, T, N \cup T, D, q, S, q)$ , що розпізнає мову КВ-граматики  $G = (N, T, P, S)$ , будеться на основі розширеного МП-автомата з одним станом  $q$ . Вхідний алфавіт автомата складається з термінальних символів  $T$ -граматики, алфавіт магазинних символів складається з термінальних та нетермінальних символів:  $Z = N \cup T$ . Початкова конфігурація визначається так:  $(q, \alpha, \varepsilon)$  – автомат перебуває в своєму єдиному стані  $q$ , зчитувальна головка знаходиться на початку ланцюжка  $\alpha \in T^*$ , стек порожній. Кінцева конфігурація автомата визначається так:  $(q, \varepsilon, S)$  – автомат перебуває в своєму єдиному стані  $q$ , зчитувальна головка знаходиться під порожнім символом (за кінцем ланцюжка), в магазині знаходиться початковий символ  $S$ .

# Висхідний МП-автомата для KB-граматики. (Працює як $LR(0)$ -аналізатор)

Функція переходів МП-автомата будується на основі правил граматики:

- 1)  $(q, A) \in D(q, \varepsilon, \gamma), A \in N, (\gamma \in (N \cup T)^*, \text{ якщо } (A \rightarrow \gamma) \in P;$
- 2)  $D(q, a, \varepsilon) = (q, a), \forall a \in T.$

Неформально роботу цього МП-автомата можна описати так:

- якщо в магазині знаходиться ланцюжок  $\gamma$ , то його можна замінити на нетермінальний символ  $A$ , не рухаючи при цьому головки зчитування, якщо в граматиці  $G$  є правило  $A \rightarrow \gamma$ ;
- якщо зчитується деякий символ  $a$  вхідного ланцюжка, то його можна помістити в магазин та зсунути головку зчитування на одну позицію праворуч.

# Створення МП-автомата для КВ-граматики.

Для КВ-граматики  $G = (N, T, P, S)$  визначимо МП-автомат  $=(\{q\}, T, T \cup N, D, q, S, \{q\})$  (з єдиним станом  $q$ ), система команд  $D$  якого сформована наступним чином: для кожного правила  $A \rightarrow \alpha$ , що належить  $P$ , в  $D$  добавляється команда  $D(q, \varepsilon, A) \rightarrow \{(q, \alpha)\}$  і (для кожного)  $\forall a \in T$  – команда  $D(q, a, a) \rightarrow \{(q, \varepsilon)\}$ .

Робота цього МП-автомата  $M$  відповідає лівому виводу.

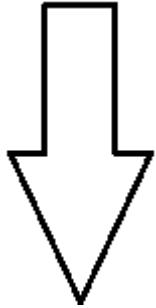
Тип синтаксичного аналізу, що виконує МП-автомат, побудований таким чином називається “**низхідним аналізом**”, або аналізом “згори донизу”, чи прогнозуючим. Під час такого аналізу дерево виводу будується, починаючи з кореня, у напрямку згори донизу, і кожний тakt можна трактувати як передвищенння чергового кроку лівого виводу.

Створення МП-автомата для КВ-граматики. Приклад:

$$G = (\{a, b\}, \{S\}, P, S),$$

$$P = \{S \rightarrow aSb | ab | \varepsilon\}.$$

$$L(M) = \{ a^n b^n, n \geq 0 \}$$



$$\begin{aligned} &= (\{q\}, \{a, b\}, \{a, b, S\}, D, q, S, \{q\}), \\ &D(q, \varepsilon, S) \rightarrow \{(q, aSb)\}, \\ &D(q, \varepsilon, S) \rightarrow \{(q, ab)\}, \\ &D(q, \varepsilon, S) \rightarrow \{(q, \varepsilon)\}, \\ &D(q, a, a) \rightarrow \{(q, \varepsilon)\}, \\ &D(q, b, b) \rightarrow \{(q, \varepsilon)\}. \end{aligned}$$

Застосовано:

$$\begin{aligned} &D(q, \varepsilon, A) \rightarrow \{(q, a)\} \text{ для } A \rightarrow a \\ &D(q, \varepsilon, A) \rightarrow \{(q, a)\} \text{ для } A \rightarrow a \\ &D(q, \varepsilon, A) \rightarrow \{(q, a)\} \text{ для } A \rightarrow a \\ &D(q, a, a) \rightarrow \{(q, \varepsilon)\} \text{ для } a \in T \\ &D(q, a, a) \rightarrow \{(q, \varepsilon)\} \text{ для } a \in T \end{aligned}$$

# Дерева виводу.

Впорядковане помічене дерево  $D$  називається *деревом виводу* (або *деревом розбору*) слова  $w$  в KB-граматиці  $G = (N, T, P, S)$ , якщо дотримано таких умов:

- (1) корінь дерева  $D$  позначено аксіомою  $S$ ;
- (2) кожен лист позначено або певним терміналом  $a \in T$ , або  $\epsilon$ ;
- (3) кожну внутрішню вершину позначено нетерміналом  $A \in N$ ;
- (4) якщо  $X$  – нетермінал, яким позначено внутрішню вершину і  $X_m \dots X_k$  – мітки її прямих потомків у вказаному порядку, то  $X \rightarrow X_m \dots X_k$  – правило з множини  $P$ ;
- (5) ланцюг, складений з відсортованих зліва направо міток листків, дорівнює  $w$ .

# Дерева виводу.

Вивід, в якому на кожному кроці здійснюється підстановка крайнього лівого нетерміналу, називається **лівосторонім**. Якщо  $S=>^*$  і в процесі лівостороннього виводу, то  $и$  – **ліва сентенціальна форма**. Аналогічно визначимо правостороній вивід. Позначимо кроки лівого (правого) виводу  $=>_1 (=>_r)$ .

# Вивід при застосуванні типів аналізаторів

Лівосторонній вивід характерний для **низхідного аналізу** при застосуванні **низхідного МП-автомата**. Такі аналізатори називають LL(k)-аналізаторами, де позначення означають:

- L Вхідний ланцюжок читається зліва-направо
- L Будується лівосторонній вивід
- (k) Кількість символів непрочитаної частини вхідного ланцюжка, що бачить аналізатор.

Правосторонній вивід характерний для **висхідного аналізу** при застосуванні **висхідного МП-автомата**. Такі аналізатори називають LR(k)-аналізаторами, де позначення означають:

- L Вхідний ланцюжок читається зліва-направо
- R Будується правосторонній вивід
- (k) Кількість символів непрочитаної частини вхідного ланцюжка, що бачить аналізатор. LR(0)-аналізатор не враховує непрочитану частину ланцюжка, а зважає лише на вмістиме магазину МП-автомата. LR(0) рідко застосовується на практиці, але є основою для алгоритмів SLR (1) і LALR (1).

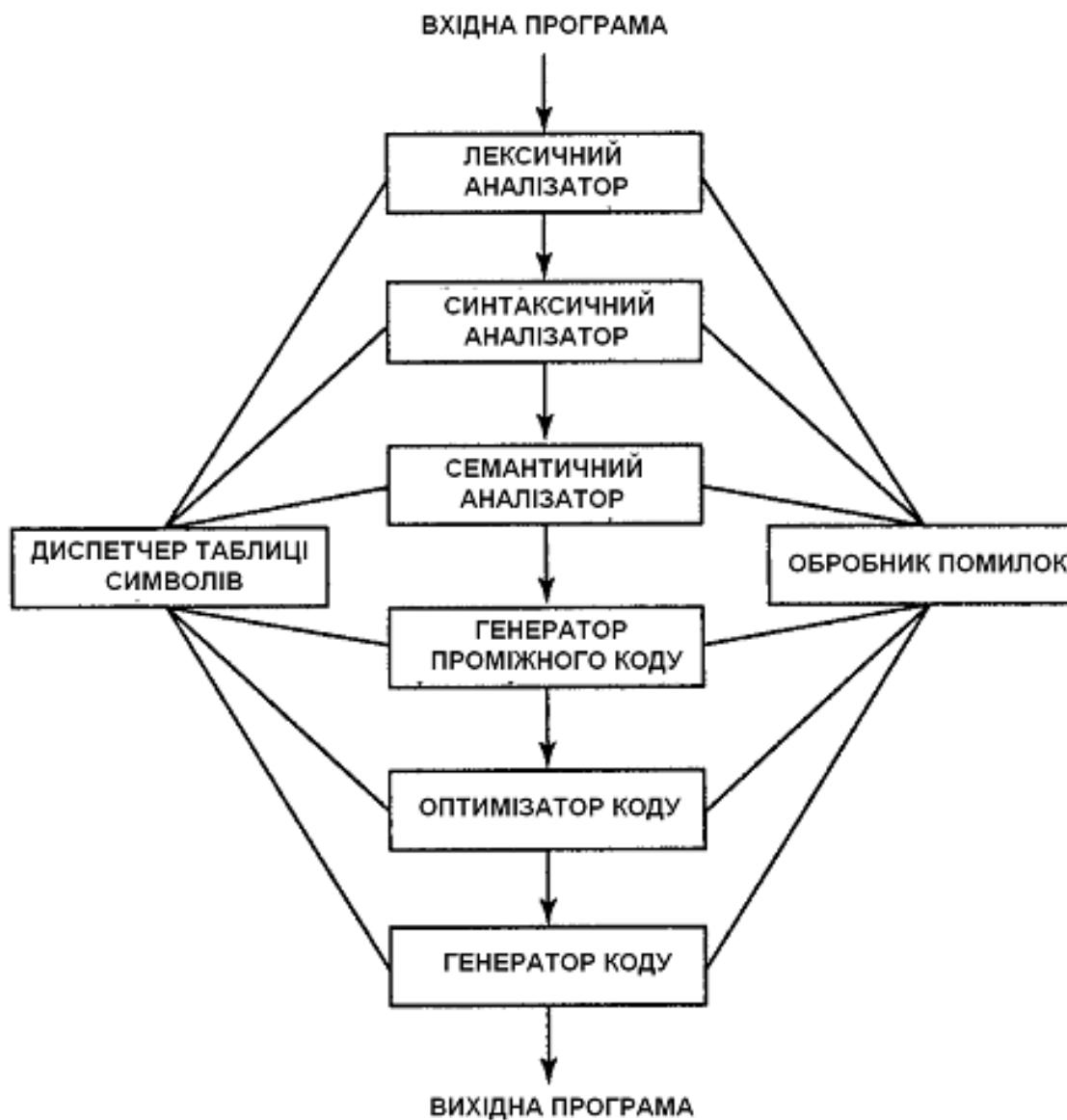
Для різних типів аналізаторів характерні відповідні підкласи КВ-граматик.

# Альтернативність дерев виводу при неоднозначності КВ-граматик.

Граматика  $G$  називається **неоднозначною**, якщо існує ланцюг  $w$ , для якого є два або більше різних дерев виводу в  $G$ .

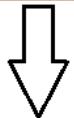
Оскільки за деревом виводу ланцюжка  $w$  з нетерміналу  $S$  однозначно будується і лівий, і правий вивід цього ланцюжка, то визначення можна сформулювати інакше:

КВ-граматика однозначна, якщо кожен ланцюжок мови, що породжується нею, має єдині лівий і правий виводи.



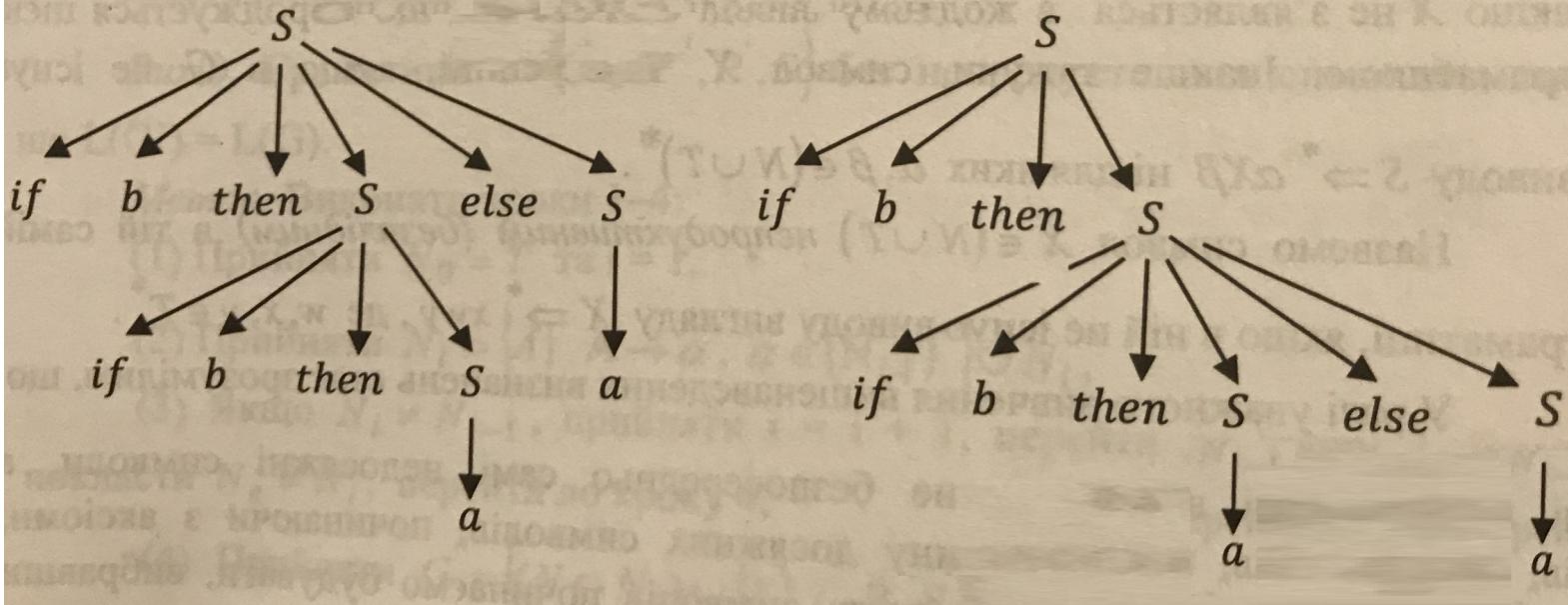
# Альтернативність дерев виводу при неоднозначності КВ-граматик. Приклад:

$$G_{if} = ( \quad T = \{a, b, if, then, else\}; \quad N = \{S\}; \quad S; \\ P = \{ S \rightarrow if \ b \ then \ S \ else \ S \mid if \ b \ then \ S \mid a \} ).$$



Визначена так граматика є неоднозначною, що видно з такого прикладу:

Ланцюжок  $if \ b \ then \ if \ b \ then \ a \ else \ a$  має два дерева виводу



## НОРМАЛЬНА ФОРМА ХОМСЬКОГО ДЛЯ КВ-ГРАМАТИК.

Граматика в нормальній формі Хомського (граматика в бінарній нормальній формі, квадратична граматика, grammar in Chomsky normal form) - контекстно-вільна граматика  $\langle N, \Sigma, S, P \rangle$ , в якій кожне правило є одним з таких трьох видів:

$$S \rightarrow \varepsilon, \quad S \text{ — аксіома}$$

$$A \rightarrow a, \quad A \in N, a \in \Sigma$$

$$A \rightarrow BC, \quad A \in N, \quad B \in N - \{S\}, C \in N - \{S\}$$

Приклад  
Граматика

$$\begin{aligned} S &\rightarrow RR; \quad S \rightarrow AB; \quad R \rightarrow RR; \quad R \rightarrow AB; \\ A &\rightarrow a; \quad B \rightarrow RB; \quad B \rightarrow b \end{aligned}$$

є граматикою в нормальній формі Хомського.

Кожна контекстно-вільна граматика еквівалентна деякій граматиці в нормальній формі Хомського.

# Перетворення КВ-граматики до нормальної форми Хомського.

Нехай дано контекстно-вільну граматику  $G = \langle N, \Sigma, S, P \rangle$ . Проведемо ряд перетворень цієї граматики так, щоб породжувана нею мова залишалася незмінною.

# Перетворення KB-граматики до нормальної форми Хомського.

(1) Якщо права частина деякого правила містить символ  $S$ , то замінимо граматику  $N, \Sigma, S, P$  на граматику

$$\langle N \cup \{S_0\}, \Sigma, S_0, P \cup \{S_0 \rightarrow S\} \rangle,$$

де  $S_0$  – нова аксіома граматики і відповідно новий нетермінал, що не належить множині  $N \cup \Sigma$ . Так ми позбуваємося випадку, коли аксіома граматики зустрічається в правих частинах правил.

(2) Замінимо у всіх правилах кожен термінальний символ  $a$  на новий нетермінальний символ  $T_a$  і додамо до множини  $P$  правила  $T_a \rightarrow a$  для всіх  $a \in \Sigma$ .

# Перетворення KB-граматики до нормальної форми Хомського.

- (3) Видалимо правила вигляду  $A \rightarrow \alpha$ , де  $|\alpha| > 2$ , замінивши кожне з них на ряд коротких правил по два нетермінали кожному (при цьому додаються нові нетермінальні символи).
- (4) Тепер видалимо всі правила вигляду  $A \rightarrow \varepsilon$ , де  $A$  не є початковим символом. Це можна зробити так.
- (а) Якщо для якихось  $A \in N$ ,  $B \in N$ ,  $\alpha, \beta \in (N \cup T)^*$ , множина  $P$  містить правила  $B \rightarrow \alpha A \beta$  і  $A \rightarrow \varepsilon$ , але не містить правила  $B \rightarrow \alpha \beta$ , то додамо це правило в  $P$ . Повторюємо цю процедуру для всіх  $\varepsilon$ -породжуючих нетерміналів.
- (б) Тепер виключимо з множини  $P$  всі правила з  $\varepsilon$ -породжуючими нетерміналами в лівій частині правил граматики. Отримана граматика породжує мову  $L - \{\varepsilon\}$ .
- (5) Якщо для будь-яких  $A, B \in N$  і  $\alpha \in (N \cup \Sigma)^*$  множина  $P$  містить правила  $A \rightarrow B$  і  $B \rightarrow \alpha$ , але не містить правила  $A \rightarrow \alpha$ , то додаємо це правило в  $P$ . Правила  $A \rightarrow B$  називаються ланцюговими. Повторюємо цю процедуру, доки можливо. Після цього виключимо з множини  $P$  всі правила вигляду  $A \rightarrow B$ .

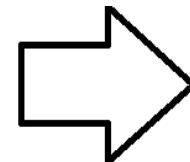
# Перетворення КВ-граматики до нормальної форми Хомського.

**Зауваження.** Перетворення КВ-граматик необхідно здійснювати **САМЕ** в **ТАКІЙ ПОСЛІДОВНОСТІ**:

- а) видалити  $\epsilon$ -продукції;
- б) видалити ланцюгові продукції;
- в) видалити некорисні символи;
- г) ввести нову аксіому, якщо аксіома попередньої граматики вживалась в правих частинах правил;
- д) замінити правила вигляду  $A \rightarrow \alpha$ , де  $|\alpha| > 2$ .

# Перетворення КВ-граматики до нормальної форми Хомського. Приклад:

$G = (\{a, b, c\}, \{S, A, B\}, P, S),$   
 $P = \{S \rightarrow ABa,$   
 $A \rightarrow aab,$   
 $B \rightarrow Ac\}.$



$G = (\{a, b, c\}, \{S, A, B, V_1, V_2, T_a, T_b, T_c\}, P, S),$   
 $P = \{S \rightarrow AV_1,$   
 $V_1 \rightarrow BT_a,$   
 $A \rightarrow T_a V_2,$   
 $V_2 \rightarrow T_a T_b,$   
 $B \rightarrow AT_c,$   
 $T_a \rightarrow a,$   
 $T_b \rightarrow b,$   
 $T_c \rightarrow c\}.$

## **Алгоритм синтаксичного аналізу Кока–Касамі–Янгера для граматик у нормальній формі Хомського**

Існує достатньо ефективний метод визначення чи належить ланцюжок мові, що задана граматикою, який оснований на ідеї “динамічного програмування”. Цей алгоритм відомий ще як СУК-алгоритм – алгоритм Кока–Янгера–Касамі.

Він використовується лише для граматик у нормальній формі Хомського.

На вхід алгоритму подається ланцюжок  $w = a_1 a_2 \dots a_n$  з  $T^*$ .

За час  $O(n^3)$  алгоритм будує таблицю, яка говорить, чи належить  $w$  мові  $L$ .

# Алгоритм Кока-Касамі-Янгера для синтаксичного аналізу.

Базис. Обчислюємо перший рядок так. Оскільки ланцюжок, який починається та закінчується в позиції  $i$ , являє собою просто термінал  $a_i$ , а граматика знаходиться в НФХ, єдиний спосіб породити  $a_i$  полягає в використанні продукції вигляду  $A \rightarrow a_i$  граматики  $G$ . Отже,  $X_{ii}$  є множиною змінних  $A$ , для яких  $A \rightarrow a_i$  — продукція  $G$ .

Індукція. Нехай потрібно обчислити  $X_{ij}$  в  $(j - i + 1)$ -му рядку, і всі множини  $X$  в нижніх рядках вже обчислені, тобто відомі для всіх подланцюжків, коротших, ніж  $a_i a_{i+1} \dots a_j$ , і зокрема, для всіх власних префіксів і суфіксів цього ланцюжка. Можна припустити, що  $j - i > 0$ , оскільки випадок  $j = i$  розглянуто в базисі. Тому будь-який вивід  $A \Rightarrow^* a_i a_{i+1} \dots a_j$  має починатися кроком  $A \rightarrow BC$ . Тоді  $B$  породжує деякий префікс рядка  $a_i a_{i+1} \dots a_j$ , скажімо,  $B \Rightarrow^* a_i a_{i+1} \dots a_k$  для деякого  $k < j$ . Відповідно,  $C$  породжує залишок  $a_{k+1} a_{k+2} \dots a_j$ , тобто  $C \Rightarrow^* a_{k+1} a_{k+2} \dots a_j$ .

Схема індексації таблиці аналізу 

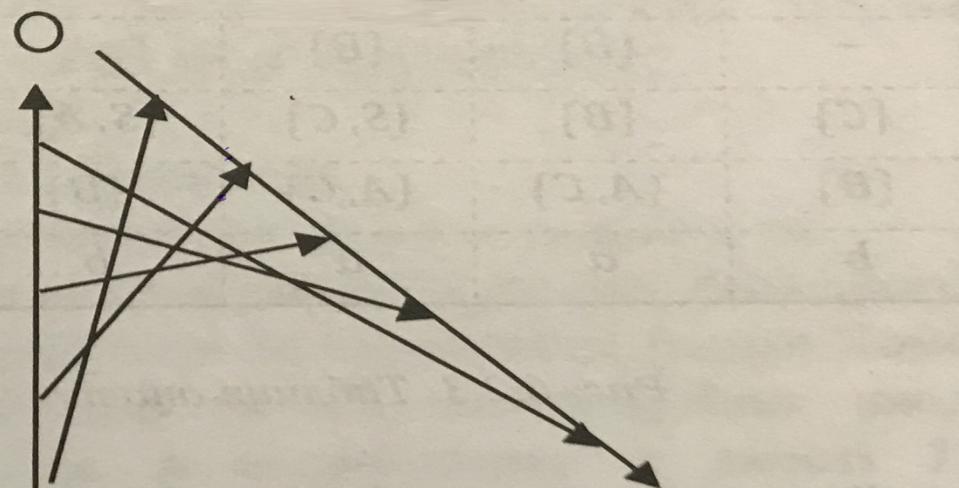
		$X_{15}$			
	$X_{14}$		$X_{25}$		
$X_{13}$		$X_{24}$		$X_{35}$	
$X_{12}$		$X_{23}$		$X_{34}$	$X_{45}$
$X_{11}$		$X_{22}$		$X_{33}$	$X_{44}$
$a_1$		$a_2$		$a_3$	$a_4$
					$a_5$

# Алгоритм Кока-Касамі-Янгера для синтаксичного аналізу.

Доходимо висновку, що для того, щоб  $A$  потрапило в  $X_{ij}$ , потрібно знайти змінні  $B$  і  $C$  і ціле  $k$ , при яких справедливі такі умови:

- 1)  $i \leq k < j$ ;
- 2)  $B$  належить  $X_{ik}$ ;
- 3)  $C$  належить  $X_{k+1,j}$ ;
- 4)  $A \rightarrow BC$  – продукція в  $G$ .

Пошук таких змінних  $A$  потребує обробки не більше  $n$  пар обчислених раніше множин:  $(X_{ii}, X_{i+1}, j)$ ,  $(X_{i+1}, X_{i+2}, j)$  і т. д. до  $(X_{i-j-1}, X_{jj})$ . Отже, ми піднімаємося по колонці, розташованій під  $X_{ij}$ , і одночасно спускаємося по діагоналі.



Алгоритм Кока-Касамі-Янгера для синтаксичного аналізу. Приклад:

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

$$P = \{ S \rightarrow AB \mid BC , \\ A \rightarrow BA \mid a , \\ B \rightarrow CC \mid b , \\ C \rightarrow AB \mid a \} .$$

Перевіримо, чи належить ланцюжок  $baaba$  мові  $L(G)$ .



Таблиця аналізу

{S, A, C}				
-	{S, A, C}			
-	{B}	{B}		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a



## Правила перетворення.

### 1. Ітерація {X} (багаторазове повторення).

- Замінюються на новий нетермінал N\_iter\_X
- Додаються правила:

$$N_{\text{iter}}_X \rightarrow X N_{\text{iter}}_X$$
$$N_{\text{iter}}_X \rightarrow \epsilon$$

## Правила перетворення.

### 2. Опціонал [X] (необов'язковий елемент).

- Замінюються на новий нетермінал  $N_{opt\_X}$
- Додаються правила:

$$N_{opt\_X} \rightarrow X$$

$$N_{opt\_X} \rightarrow \epsilon$$

## Правила перетворення.

### 3. Альтернатива $X \mid Y$ (вибір одного з варіантів).

- Замінююється на окремі правила:

$N \rightarrow X$

$N \rightarrow Y$

### 4. Конкатенація (послідовність елементів) записується як $\epsilon$ (кома з EBNF відкидається).

<b><u>Варіант 0</u></b> EBNF: $S = \{A \mid B\}, "c"$	<b><u>Варіант 1</u></b> EBNF: $S = "a", [B \mid C], "d"$	<b><u>Варіант 2</u></b> EBNF: $S = \{A \mid "b"\}, C$
<b><u>Варіант 5</u></b> EBNF: $S = \{A \mid [B]\}, "c"$	<b><u>Варіант 6</u></b> EBNF: $S = "x" \mid "y" \mid \{A\}, "z"$	<b><u>Варіант 7</u></b> EBNF: $S = [A \mid B \mid C], \{D\}, "end"$

<b><u>Варіант 3</u></b> EBNF: $S = "start", \{A \mid B \mid C\}, "end"$	<b><u>Варіант 4</u></b> EBNF: $S = [A \mid B], \{C\}, "finish"$
<b><u>Варіант 8</u></b> EBNF: $S = \{A \mid [B] \mid C\}, D$	<b><u>Варіант 9</u></b> EBNF: $S = "begin", \{A \mid B \mid [C]\}, "end"$

# Приклад. Варіанти граматики для EBNF: $S = \{A \mid B\}, "c"$

Граматика  $G_1$

$G_1 = (N_1, T_1, P_1, S)$

$N_1 = \{S, S_{\_iter\_AB}\}$

$T_1 = \{A, B, "c"\}$

$P_1 = \{$

$S \rightarrow S_{\_iter\_AB} "c",$

$S_{\_iter\_AB} \rightarrow A S_{\_iter\_AB},$

$S_{\_iter\_AB} \rightarrow B S_{\_iter\_AB},$

$S_{\_iter\_AB} \rightarrow \epsilon$

}

Граматика  $G_2$  (з проміжним нетерміналом  $C$ )

text

$G_2 = (N_2, T_2, P_2, S)$

$N_2 = \{S, Iter\_C, C\}$

$T_2 = \{A, B, "c"\}$

$P_2 = \{$

$S \rightarrow Iter\_C "c",$

$C \rightarrow A,$

$C \rightarrow B,$

$Iter\_C \rightarrow C Iter\_C,$

$Iter\_C \rightarrow \epsilon$

}