

# Теория автоматов и формальных языков

## Контекстно-свободные языки

**Лектор:** Екатерина Вербицкая

НИУ-ВШЭ

26 сентября 2022

## В предыдущей серии

- Регулярные выражения, регулярные грамматики и конечные автоматы задают класс регулярных языков
- Класс регулярных языков замкнут относительно теоретико-множественных операций, конкатенации, итерации, гомоморфизма цепочек
- Определение принадлежности слова языку осуществляется за  $O(n)$  операций
- Однако класс регулярных языков достаточно узок, ни один используемый в промышленности язык программирования не является регулярным
  - ▶ Лемма о накачке для доказательства нерегулярности языка
  - ▶ Язык правильных скобочных последовательностей, язык палиндромов не являются регулярными

# Контекстно-свободная грамматика

Четверка  $\langle V_T, V_N, P, S \rangle$

- $V_T$  — алфавит терминальных символов (терминалов)
- $V_N$  — алфавит нетерминальных символов (нетерминалов)
  - ▶  $V_T \cap V_N = \emptyset$
  - ▶  $V ::= V_T \cup V_N$
- $P$  — конечное множество правил вида  $A \rightarrow \alpha$ 
  - ▶  $A \in V_N$
  - ▶  $\alpha \in V^*$
- $S$  — начальный нетерминал грамматики,  $S \in V_N$

Пример: арифметические выражения

$$E \rightarrow E + E \mid E * E \mid N$$

$$N \rightarrow 0 \mid 1 \mid \dots \mid 9$$

# Вывод в грамматике

- **Отношение выводимости:**

$$\forall \alpha, \gamma, \delta \in V^*, A \in V_N : A \rightarrow \alpha \in P : \gamma A \delta \Rightarrow \gamma \alpha \delta$$

- **Вывод** — транзитивное, рефлексивное замыкание отношения выводимости ( $\overset{*}{\Rightarrow}, \overset{+}{\Rightarrow}, \overset{k}{\Rightarrow}$ )
- **Левосторонний (правосторонний) вывод** — на каждом шаге заменяем самый левый (правый) нетерминал
  - ▶ Если не специфицируется, подразумевается левосторонний вывод
- По сути, правила грамматики рассматриваются как правила переписывания

## Пример вывода

Построим левосторонний вывод цепочки  $2 + 3 * 4$  в грамматике

$$\langle \{0, 1, \dots, 9, +, *\}, \{E, N\}, P, E \rangle$$

$$E \rightarrow E + N \mid E * N \mid N$$

$$N \rightarrow 0 \mid 1 \mid \dots \mid 9$$

$$\underline{E} \Rightarrow \underline{E} * N \Rightarrow \underline{E} + N * N \Rightarrow \underline{N} + N * N \Rightarrow 2 + N * N \stackrel{2}{\Rightarrow} 2 + 3 * 4$$

# Существование левостороннего вывода

## Теорема

Если для цепочки  $\omega$  существует некоторый вывод  $S \xRightarrow{*} \omega$ ,  
то существует и левосторонний вывод для этой цепочки  $S \xRightarrow{*}_l \omega$

## Доказательство.

Докажем более общее утверждение:

если существует  $A \xRightarrow{*} \omega$ , то существует  $A \xRightarrow{*}_l \omega$ , где  $A \in V_N$ .

Доказываем по индукции по длине вывода  $k$

$k = 1$  :  $A \Rightarrow \omega$  — тривиально.

$k \mapsto k + 1$  :  $\triangleleft A \Rightarrow \alpha \xRightarrow{*} \omega$ .

Обозначим  $\alpha = B_1 B_2 \dots B_m \xRightarrow{*} \omega_1 \omega_2 \dots \omega_m = \omega$ ;  $\forall i : B_i \xRightarrow{l_i} \omega_i, l_i \leq k$

По индукционному предположению  $\forall i : B_i \xRightarrow{*}_l \omega_i$

$\Rightarrow$  :  $A \Rightarrow B_1 B_2 \dots B_m \xRightarrow{*}_l \omega_1 B_2 \dots B_m \xRightarrow{*}_l \omega$  — левосторонний вывод



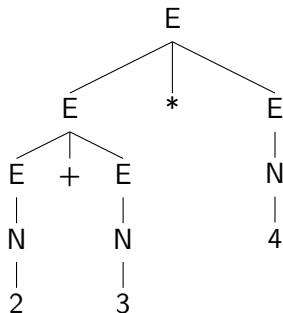
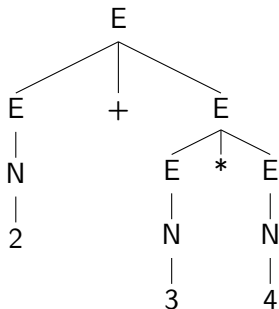
# Единственность вывода

Не всегда (левосторонний) вывод единственен:

2 вывода строки  $2 + 3 * 4$

$$E \rightarrow E + E \mid E * E \mid N$$

$$N \rightarrow 0 \mid 1 \mid \dots \mid 9$$



# Однозначность грамматики

- Грамматика называется **однозначной**, если для *любого* слова языка существует *единственный* (левосторонний) вывод
- Грамматика называется **неоднозначной**, если *существует* слово языка, такое что для него *существует несколько* (левосторонних) выводов



# Однозначность грамматики

- Грамматика называется **однозначной**, если для *любого* слова языка существует *единственный* (левосторонний) вывод
- Грамматика называется **неоднозначной**, если *существует* слово языка, такое что для него *существует несколько* (левосторонних) выводов
- По однозначной грамматике можно тривиальным образом построить неоднозначную: продублировать правило

$$S \rightarrow A$$

$$A \rightarrow a$$

$$S \rightarrow A \mid B$$

$$A \rightarrow a$$

$$B \rightarrow a$$

- Не существует общего алгоритма преобразования неоднозначной грамматики в однозначную

# Примеры однозначной и неоднозначной грамматики

Неоднозначная грамматика

$$E \rightarrow E + E \mid E * E \mid N$$

$$N \rightarrow 0 \mid 1 \mid \dots \mid 9$$

Однозначная грамматика

$$E \rightarrow E + N \mid E * N \mid N$$

$$N \rightarrow 0 \mid 1 \mid \dots \mid 9$$

# Проверка однозначности грамматики неразрешима

## Теорема

*Не существует алгоритма, определяющего по произвольной грамматике, что она является неоднозначной*

## Доказательство.

Сведение решения проблемы соответствий Поста к нашей задаче



## Проблема соответствий Поста

Даны списки  $A = (a_1, \dots, a_n)$  и  $B = (b_1, \dots, b_n)$ , где  $\forall i : a_i, b_i \in \Sigma^*$ .  
Существует ли непустая последовательность  $(i_1, \dots, i_k)$ ,  
удовлетворяющая условию  $a_{i_1} \dots a_{i_k} = b_{i_1} \dots b_{i_k}$ , где  $\forall j : 1 \leq i_j \leq n$ ?

# Проверка однозначности грамматики неразрешима

## Теорема

*Не существует алгоритма, определяющего по произвольной грамматике, что она является неоднозначной*

## Доказательство.

◁ алфавит  $\Sigma = \{a_1, \dots, a_n, b_1, \dots, b_n\} \sqcup \{z_1, \dots, z_n\}$ , где  $a_i, b_i$  из ПСП

◁ грамматику:

$$S \rightarrow A \mid B$$

$$A \rightarrow a_i A z_i \mid \varepsilon$$

$$B \rightarrow b_i B z_i \mid \varepsilon$$

Если грамматика неоднозначна, то существует выводимое слово вида:

$$a_{i_1} a_{i_2} \dots a_{i_k} z_{i_k} z_{i_{k-1}} \dots z_{i_1} = \omega = b_{i_1} b_{i_2} \dots b_{i_k} z_{i_k} z_{i_{k-1}} \dots z_{i_1}$$

Если бы умели решать ПСП, то мы могли бы проверить грамматику на однозначность, но ПСП неразрешима, а значит и проверка на однозначность неразрешима



- Язык называется **контекстно-свободным**, если для него *существует* контекстно-свободная грамматика
- Язык, задаваемый КС грамматикой  $\langle V_T, V_N, P, S \rangle$ :  
 $\{\omega \in V_T^* \mid S \xRightarrow{*} \omega\}$
- КС язык называется **существенно неоднозначным**, если для него не существует однозначной грамматики
  - ▶  $\{0^a 1^b 2^c \mid a = b \text{ либо } b = c\}$     доказательство в книге Ахо Ульмана

## Теорема

*Существует алгоритм, определяющий, является ли язык, порождаемый КС грамматикой, пустым*

## Доказательство.

Для доказательства потребуется следующая лемма



# Лемма

## Теорема

*Если в данной грамматике выводится некоторая цепочка, то существует цепочка, дерево вывода которой не содержит ветвей длиннее  $m$ , где  $m$  — количество нетерминалов грамматики*

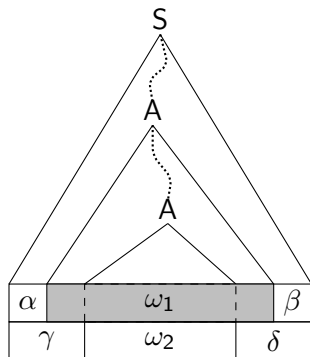
## Доказательство.

Рассмотрим дерево вывода цепочки  $\omega$ . Если в нем есть 2 узла, соответствующих одному нетерминалу  $A$ , обозначим их  $n_1$  и  $n_2$ . Предположим,  $n_1$  расположен ближе к корню дерева, чем  $n_2$ ;  
 $S \xRightarrow{*} \alpha A_{n_1} \beta \xRightarrow{*} \alpha \omega_1 \beta$ ;  $S \xRightarrow{*} \gamma A_{n_2} \delta \xRightarrow{*} \gamma \omega_2 \delta$ ;  $\omega_2$  является подцепочкой  $\omega_1$ .  
Заменим в изначальном дереве узел  $n_1$  на  $n_2$ . Полученное дерево является деревом вывода  $\alpha \omega_2 \beta$ . Повторяем процесс замены одинаковых нетерминалов до тех пор, пока в дереве не останутся только уникальные нетерминалы.  
В полученном дереве не может быть ветвей длины большей, чем  $m$ . По построению оно является деревом вывода. □

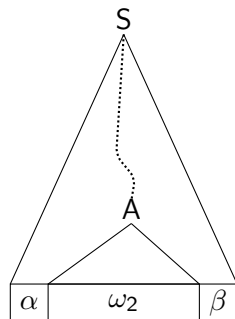
# Лемма

## Теорема

Если в данной грамматике выводится некоторая цепочка, то существует цепочка, дерево вывода которой не содержит ветвей длиннее  $m$ , где  $m$  — количество нетерминалов грамматики



$\Rightarrow$





# Алгоритм проверки пустоты КС языка

## Доказательство.

Строим коллекцию деревьев, представляющих вывод в грамматике.

- 1 Инициализируем коллекцию деревом из одного узла  $S$
- 2 Добавляем в коллекцию дерево, полученное применением единственного правила грамматики из какого-нибудь дерева из коллекции, если его в нем еще нет, и самая длинная ветвь не длиннее  $m$
- 3 Если после окончания построения коллекции в ней существует дерево, являющееся деревом вывода некоторой цепочки терминалов, значит, язык не пуст



# Упрощение КС грамматики: удаление непродуктивных нетерминалов

**Продуктивный нетерминал:** нетерминал, для которого существует цепочка терминалов, выводимая из него ( $\exists \omega \in V_T^* : A \xRightarrow{*} \omega$ )

**Непродуктивный нетерминал:** нетерминал, не являющийся продуктивным

# Упрощение КС грамматики: удаление непродуктивных нетерминалов

## Теорема

Для любой КС грамматики  $G = \langle V_T, V_N, P, S \rangle : L(G) \neq \emptyset$  можно построить эквивалентную грамматику, каждый нетерминал которой продуктивен

## Доказательство.

Удаляем из грамматики все нетерминалы  $A : L(A) = \emptyset$ , а также правила, использующие их.

Полученную грамматику обозначаем  $G_1$ .

Докажем, что  $L(G) = L(G_1)$ .

Очевидно,  $L(G_1) \subseteq L(G)$ .

Докажем от противного, что  $L(G) \subseteq L(G_1)$ .

Предположим, что  $\exists \omega \in L(G)$ , но  $\omega \notin L(G_1)$ . Тогда  $S \xRightarrow{*} \alpha_1 A \alpha_2 \xRightarrow{*} \omega$ , где  $A \in V_N \setminus V_{N_1}$ , но тогда  $\exists \gamma \in V_T^* : A \xRightarrow{*} \gamma$ . Противоречие □

# Упрощение КС грамматики: приведение

## Теорема

*Для любой КС грамматики, порождающей непустой язык, можно построить эквивалентную, для каждого нетерминала  $A$  которой существует вывод вида  $S \xRightarrow{*} \omega_1 A \omega_3 \xRightarrow{*} \omega_1 \omega_2 \omega_3, \omega_i \in V_T^*$*

## Доказательство.

Будем рассматривать грамматику без непродуктивных нетерминалов  $G_1 = \langle V_{N_1}, V_T, P_1, S \rangle$ .

Верно: если существует  $S \xRightarrow{*} \alpha_1 A \alpha_3, \alpha_i \in V^*$ , то  $S \xRightarrow{*} \alpha_1 A \alpha_3 \xRightarrow{*} \omega_1 A \omega_3 \xRightarrow{*} \omega_1 \omega_2 \omega_3, \omega_i \in V_T^*$

Строим множество нетерминалов, встречающихся в выводах: добавляем сначала  $S$ , потом добавляем нетерминалы, встречающиеся в правой части правил для нетерминалов из множества. Завершаем процесс, когда больше ничего не добавить. Обозначаем полученное множество  $V_{N_2}$ , удаляем все правила грамматики, содержащие нетерминалы из  $V_{N_1} \setminus V_{N_2}$

## Упрощение КС грамматики: приведение

### Доказательство.

Получили грамматику  $G_2 = \langle V_{N_2}, V_T, P_2, S \rangle$ .

Докажем:  $L(G_2) = L(G_1)$

$L(G_2) \subseteq L(G_1)$ , так как  $P_2 \subseteq P_1$

Докажем:  $L(G_1) \subseteq L(G_2)$ . Пусть  $S \xRightarrow[G_1]{*} \omega$ . Все нетерминалы, встречающиеся в этом выводе содержатся в  $V_{N_2}$ , соответственно используются только правила из  $P_2 \Rightarrow S \xRightarrow[G_2]{*} \omega$

Так как все нетерминалы  $V_{N_2}$  продуктивны, то  $S \xRightarrow{*} \omega_1 A \omega_3 \xRightarrow{*} \omega_1 \omega_2 \omega_3, \omega_i \in V_T^*$



Грамматика  $G_2$  называется **приведенной**, ее нетерминалы — **достижимыми**

Недостижимые и непродуктивные нетерминалы называются **бесполезными**

## Упрощение КС грамматики: удаление цепных правил

Правило называется **цепным**, если оно имеет вид  $A \rightarrow B$ ;  $A, B \in V_N$ .

### Теорема

Для любой КС грамматики  $G = \langle V_N, V_T, P, S \rangle$  можно построить эквивалентную, не содержащую цепных правил

### Доказательство.

Строим новое множество правил  $P_1$ .

Включаем в него все нецепные правила  $P$ .

Затем добавляем в  $P_1$  правила вида  $A \rightarrow \alpha$ , если  $A \rightarrow B$ , где  $A, B \in V_N$  и  $B \rightarrow \alpha$  — нецепное правило из  $P$ .

Замечание: достаточно проверять только цепные выводы длины меньшей, чем  $|V_N|$

Обозначим полученную грамматику за  $G_1 = \langle V_N, V_T, P_1, S \rangle$ , докажем  $L(G_1) = L(G)$  □

# Упрощение КС грамматики: удаление цепных правил

## Доказательство.

Очевидно  $L(G_1) \subseteq L(G)$

Покажем  $L(G) \subseteq L(G_1)$ . Пусть  $\omega \in L(G)$ . Рассмотрим левосторонний вывод  $S \xRightarrow{G} \alpha_0 \xRightarrow{G} \alpha_1 \xRightarrow{G} \dots \xRightarrow{G} \alpha_n = \omega$ .

Предположим  $\alpha_i \xRightarrow{G} \alpha_{i+1}$  — первый шаг, выполняемый посредством цепного правила в выводе;  $\forall k \in [i..j] : \alpha_k \xRightarrow{G} \alpha_{k+1}$  — посредством цепного правила;  $\alpha_j \xRightarrow{G} \alpha_{j+1}$  — посредством нецепного правила

Тогда  $|\alpha_i| = |\alpha_{i+1}| = \dots = |\alpha_j|$ , и на каждом шаге заменяется один и тот же нетерминал. Тогда  $\alpha_i \xRightarrow{G_1} \alpha_{j+1}$  посредством правила из  $P_1 \setminus P$ , следовательно  $\omega \in L(G_1)$  □

# Нормальная форма Хомского

КС грамматика находится в **нормальной форме Хомского**, если все ее правила имеют вид:

$$A \rightarrow BC$$

$$A, B, C \in V_N$$

$$A \rightarrow a$$

$$A \in V_N, a \in V_T$$

$$S \rightarrow \varepsilon$$

$S$  — стартовый нетерминал



# Нормальная форма Хомского

## Теорема

*Для любой КС грамматики можно построить эквивалентную в нормальной форме Хомского*

- ① Удаляем непродуктивные нетерминалы
- ① Удаляем цепные правила. Теперь  $\forall A \rightarrow B : B \in V_T$
- ② Заменяем каждое правило  $A \rightarrow B_1 B_2 \dots B_n$  на  $A \rightarrow C_1 C_2 \dots C_n$ 
  - ▶ Если  $B_i \in V_N$ ,  $C_i = B_i$ ,
  - ▶ Если  $B_i \in V_T$ , добавляем также правило  $C_i \rightarrow B_i$ ,
- ③ Заменяем правило  $A \rightarrow C_1 C_2 \dots C_n$  на множество правил:

$$A \rightarrow C_1 D_1$$

$$D_1 \rightarrow C_2 D_2$$

...

$$D_{n-3} \rightarrow C_{n-2} D_{n-2}$$

$$D_{n-2} \rightarrow C_{n-1} C_n$$

Полученная грамматика находится в НФХ и эквивалентна данной

## Пример приведения в НФХ

$G = \langle \{S, A, B\}, \{a, b\}, P, S \rangle$ , где  $P$  :

$$S \rightarrow bA \mid aB$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

- $S \rightarrow bA \Rightarrow S \rightarrow C_1A; C_1 \rightarrow b$
- $S \rightarrow aB \Rightarrow S \rightarrow C_2B; C_2 \rightarrow a$
- $A \rightarrow aS \Rightarrow A \rightarrow C_3S; C_3 \rightarrow a$
- $A \rightarrow bAA \Rightarrow A \rightarrow C_4D_1; C_4 \rightarrow b; D_1 \rightarrow AA$
- $B \rightarrow bS \Rightarrow B \rightarrow C_5S; C_5 \rightarrow b$
- $B \rightarrow aBB \Rightarrow B \rightarrow C_6D_2; C_6 \rightarrow a; D_2 \rightarrow BB$

## Еще немного упростим

$$S \rightarrow bA \mid aB$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

$$S \rightarrow C_bA \mid C_aB$$

$$A \rightarrow a \mid C_aS \mid C_bD_1$$

$$B \rightarrow b \mid C_bS \mid C_aD_2$$

$$D_1 \rightarrow AA$$

$$D_2 \rightarrow BB$$

$$C_a \rightarrow a$$

$$C_b \rightarrow b$$

# Алгоритм приведения в НФХ

- 1 Удалить стартовый нетерминал из правых частей правил
  - ▶ добавляется новое правило  $S_0 \rightarrow S$ ,  $S_0 \notin V_N$ ,  $S_0$  делается новым стартовым
- 2 Избавиться от неодинокных терминалов в правых частях
  - ▶ новое правило  $C_c \rightarrow c$
- 3 Удалить длинные правила (длины больше 2)
- 4 Удалить непродуктивные правила ( $\varepsilon$ -правила)
  - ▶ Если  $A \rightarrow \varepsilon$ , то  $A$  —  $\varepsilon$ -порождающий нетерминал
  - ▶ Если  $A \rightarrow X_1 X_2 \dots X_n$ ,  $\forall i : X_i$  —  $\varepsilon$ -порождающий, то  $A$  —  $\varepsilon$ -порождающий нетерминал
  - ▶ Заменяем  $A \rightarrow X_1 X_2 \dots X_n$  на множество правил, где каждый  $X_i$  опущен во всех возможных комбинациях, удаляем  $\varepsilon$ -правила
  - ▶  $A \rightarrow X_1 X_2 X_3 \Rightarrow A \rightarrow X_1 X_2 X_3 \mid X_2 X_3 \mid X_1 X_3 \mid X_1 X_2 \mid X_3 \mid X_2 \mid X_1 \mid \varepsilon$
- 5 Удалить цепные правила
  - ▶ Для каждой пары правил  $A \rightarrow B$ ;  $B \rightarrow X_1 X_2 \dots X_n$  добавить правило  $A \rightarrow X_1 X_2 \dots X_n$ , цепное правило удалить

# Порядок действий при приведении в НФХ

Порядок **важен!**

- ❶ Удалить стартовый нетерминал из правых частей правил
  - ❷ Избавиться от неодинокных терминалов в правых частях
  - ❸ Удалить длинные правила (длины больше 2)
  - ❹ Удалить непродуктивные правила ( $\epsilon$ -правила)
  - ❺ Удалить цепные правила
- 1 шаг порождает новые цепные правила, поэтому его нельзя выполнять после 5 шага
  - Если выполнить 4 шаг перед 3 шагом, то произойдет экспоненциальный взрыв грамматики
  - 5 шаг приводит к квадратичному возрастанию размера грамматики
  - Наиболее эффективны порядки 1, 2, 3, 4, 5 и 1, 3, 4, 5, 2

# Увеличение размера грамматики при нормализации

Порядок **важен!**

- ❶ Удалить стартовый нетерминал из правых частей правил
  - ▶ Увеличение на 1
- ❷ Избавиться от неодинокных терминалов в правых частях
  - ▶ Увеличение на  $|V_T|$  правил
- ❸ Удалить длинные правила (длины больше 2)
  - ▶ Увеличение не более, чем в 2 раза (для правил длины  $k \geq 3$  порождается  $k - 1$  новых правил)
- ❹ Удалить непродуктивные правила ( $\epsilon$ -правила)
  - ▶ Увеличение не более, чем в 3 раза
- ❺ Удалить цепные правила
  - ▶ Увеличение не более, чем в  $O(n^2)$  (цепных правил не больше  $n^2$ , где  $n$  — число нетерминалов)

Итого: **полиномиальное** увеличение размеров грамматики при правильном порядке действий

# Задача распознавания

Построить алгоритм\*, который определяет, принадлежит ли строка данному языку или нет.

```
recognizer :: String -> Grammar -> Bool
```

---

\*Алгоритм обязан завершаться

Построить алгоритм\*, который определяет, принадлежит ли строка данному языку или нет, строит дерево вывода или сообщает об ошибке.

```
parser :: String -> Grammar -> (DerivationTree | SyntaxError)
```

---

\*Алгоритм обязан завершаться



# Синтаксический анализ: алгоритм Кока-Янгера-Касами (Cocke-Younger-Kasami algorithm, CYK)

Что значит  $A \rightarrow a$ ?

# Синтаксический анализ: алгоритм Кока-Янгера-Касами (Cocke-Younger-Kasami algorithm, CYK)

Что значит  $A \rightarrow a$ ?

$$A \Rightarrow a \xRightarrow{*} \omega \Leftrightarrow \omega = a$$

# Синтаксический анализ: алгоритм Кока-Янгера-Касами (Cocke-Younger-Kasami algorithm)

Что значит  $A \rightarrow BC$ ?

# Синтаксический анализ: алгоритм Кока-Янгера-Касами (Cocke-Younger-Kasami algorithm)

Что значит  $A \rightarrow BC$ ?

$$A \Rightarrow BC \xRightarrow{*} \omega \Leftrightarrow \begin{cases} \exists \omega_1, \omega_2 : \omega = \omega_1 \omega_2 \\ B \xRightarrow{*} \omega_1 \\ C \xRightarrow{*} \omega_2 \end{cases}$$

# Синтаксический анализ: алгоритм Кока-Янгера-Касами (Cocke-Younger-Kasami algorithm)

Что значит  $A \rightarrow BC$ ?

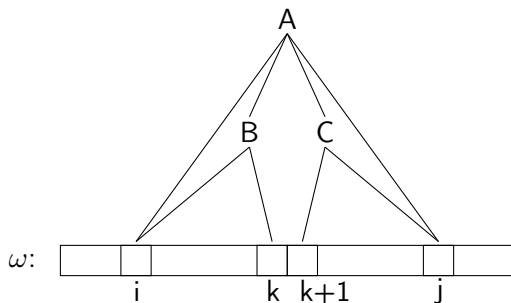
$$A \Rightarrow BC \xRightarrow{*} \omega \Leftrightarrow \begin{cases} \exists \omega_1, \omega_2 : \omega = \omega_1 \omega_2 \\ B \xRightarrow{*} \omega_1 \\ C \xRightarrow{*} \omega_2 \end{cases}$$

Или:

$$A \Rightarrow BC \xRightarrow{*} \omega \Leftrightarrow \exists k \in [0 \dots |\omega|] : \begin{cases} B \xRightarrow{*} \omega[0 \dots k] \\ C \xRightarrow{*} \omega[k + 1 \dots |\omega|] \end{cases}$$

# Синтаксический анализ: алгоритм Кока-Янгера-Касами (Cocke-Younger-Kasami algorithm, CYK)

- Алгоритм синтаксического анализа, работающий с грамматиками в НФХ
- Динамическое программирование



- Дано: строка  $\omega$  длины  $n$ , грамматика  $G = \langle V_T, V_N, P, S \rangle$  в НФХ
- Используем трехмерный массив  $d$  булевых значений размером  $|V_N| \times n \times n$ ,  $d[A][i][j] = \text{true} \Leftrightarrow A \xRightarrow{*} \omega[i \dots j]$
- Инициализация:  $i = j$ 
  - ▶  $d[A][i][i] = \text{true}$ , если в грамматике есть правило  $A \rightarrow \omega[i]$
  - ▶  $d[A][i][i] = \text{false}$ , иначе
- Динамика. Предполагаем,  $d$  построен для всех нетерминалов и пар  $\{(i', j') \mid j' - i' < m\}$ 
  - ▶  $d[A][i][j] = \bigvee_{A \rightarrow BC} \bigvee_{k=i}^j d[B][i][k] \wedge d[C][k+1][j]$
- В конце работы алгоритма в  $d[S][1][n]$  записан ответ, выводится ли  $\omega$  в данной грамматике