Теория автоматов и формальных языков
Контекстно-свободные языки

Лектор: Екатерина Вербицкая

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

3 ноября 2017г.

В предыдущей серии

- Контекстно-свободные грамматики (все правила имеют вид $A \to \alpha$)
- КС языки и разрешимость проверки пустоты
- Нормальная форма Хомского
- Алгоритм CYK

В предыдущей серии: НФХ

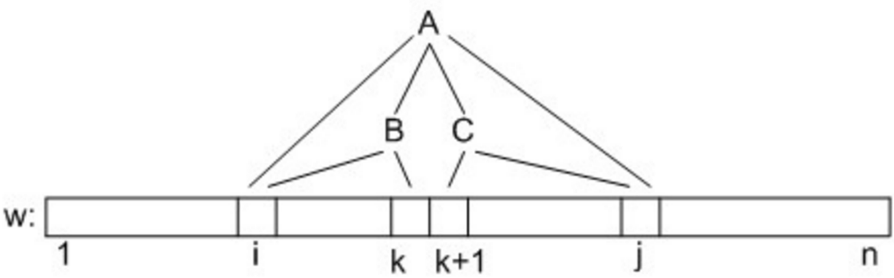
КС грамматика находится в **нормальной форме Хомского**, если все ее правила имеют вид:

- $A \to BC$, где $A, B, C \in V_N$
- $A \to a$, где $A \in V_N, a \in V_T$
- $S \to \varepsilon$, если в языке есть пустое слово, где S — стартовый нетерминал

1. Удалить стартовый нетерминал из правых частей правил
2. Избавиться от неодиночных терминалов в правых частях
3. Удалить длинные правила (длины больше 2)
4. Удалить непродуктивные правила (ε-правила)
5. Удалить цепные правила

В предыдущей серии: CYK

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

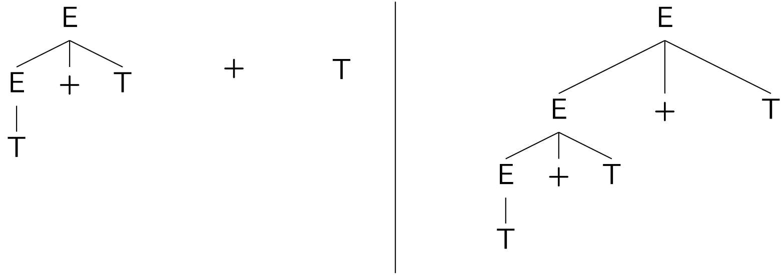


В предыдущей серии: CYK

- Дано: строка ω длины n, грамматика $G = \langle V_T, V_N, P, S \rangle$ в НФХ
- Используем трехмерный массив d булевых значений размером $|V_N| \times n \times n$, $d[A][i][j] = true \Leftrightarrow A \stackrel{*}{\Rightarrow} \omega[i \dots j]$
- Инициализация: $i = j$
  - $d[A][i][i] = true$, если в грамматике есть правило $A \rightarrow \omega[i]$
  - $d[A][i][i] = false$, иначе
- Динамика. Предполагаем, d построен для всех нетерминалов и пар $\{(i', j') \,|\, j' - i' < m\}$
  - $d[A][i][j] = \bigvee_{A \rightarrow BC} \bigvee_{k=i}^{j-1} d[B][i][k] \wedge d[C][k+1][j]$
- В конце работы алгоритма в $d[S][0][n]$ записан ответ, выводится ли ω в данной грамматике

CYK — алгоритм восходящего анализа

Восходящий анализ: начинаем с символов входной строки, строим дерево вывода до стартового нетерминала



Восходящий анализ контринтуинтивен (особенно при диагностике ошибок)

Нисходящий синтаксический анализ

- Top-down parsing
- Начинаем разбирать со стартового нетерминала, применяем правила грамматики, пока не получим строку
  - С откатом ([full] backtracking)
  - Без отката (without backtracking)

Нисходящий синтаксический анализ с откатом

- Метод грубой силы, bruteforce
- Перебираем все возможные варианты разбора, если что-то пошло не так — возвращаемся к началу и пробуем снова

Bruteforce parsing: пример

$$
\begin{array}{lcccl}
S & \to & aAd & | & aB \\
A & \to & b & | & c \\
B & \to & ccd & | & ddc
\end{array}
$$

$\omega = addc$

Bruteforce parsing: пример

$$
\begin{array}{rclcl}
S & \to & aAd & | & aB \\
A & \to & b & | & c \\
B & \to & ccd & | & ddc
\end{array}
$$

$\omega = addc$

S

Bruteforce parsing: пример

$$\begin{array}{lcccc} S & \rightarrow & aAd & \mid & aB \\ A & \rightarrow & b & \mid & c \\ B & \rightarrow & ccd & \mid & ddc \end{array}$$

$\omega = addc$

$S \Rightarrow aAd$

Bruteforce parsing: пример

$$
\begin{array}{lcccc}
S & \to & aAd & | & aB \\
A & \to & b & | & c \\
B & \to & ccd & | & ddc
\end{array}
$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$

Bruteforce parsing: пример

$$
\begin{array}{lcccl}
S & \to & aAd & | & aB \\
A & \to & b & | & c \\
B & \to & ccd & | & ddc
\end{array}
$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

Bruteforce parsing: пример

$$
\begin{array}{lcccc}
S & \rightarrow & aAd & | & aB \\
A & \rightarrow & b & | & c \\
B & \rightarrow & ccd & | & ddc
\end{array}
$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd$

Bruteforce parsing: пример

$$
\begin{array}{rcccl}
S & \to & aAd & | & aB \\
A & \to & b & | & c \\
B & \to & ccd & | & ddc
\end{array}
$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd \Rightarrow acd$

Bruteforce parsing: пример

$$
\begin{array}{rcccl}
S & \to & aAd & \mid & aB \\
A & \to & b & \mid & c \\
B & \to & ccd & \mid & ddc
\end{array}
$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd \Rightarrow acd$ — не подходит, откатываемся

Bruteforce parsing: пример

$$
\begin{array}{lcccl}
S & \rightarrow & aAd & \mid & aB \\
A & \rightarrow & b & \mid & c \\
B & \rightarrow & ccd & \mid & ddc
\end{array}
$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd \Rightarrow acd$ — не подходит, откатываемся

S

Bruteforce parsing: пример

$$
\begin{array}{rcccl}
S & \to & aAd & \mid & aB \\
A & \to & b & \mid & c \\
B & \to & ccd & \mid & ddc
\end{array}
$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd \Rightarrow acd$ — не подходит, откатываемся

$S \Rightarrow aB$

Bruteforce parsing: пример

$$
\begin{array}{lcccc}
S & \rightarrow & aAd & | & aB \\
A & \rightarrow & b & | & c \\
B & \rightarrow & ccd & | & ddc
\end{array}
$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd \Rightarrow acd$ — не подходит, откатываемся

$S \Rightarrow aB \Rightarrow accd$

Bruteforce parsing: пример

$$\begin{array}{rcccc} S & \rightarrow & aAd & \mid & aB \\ A & \rightarrow & b & \mid & c \\ B & \rightarrow & ccd & \mid & ddc \end{array}$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd \Rightarrow acd$ — не подходит, откатываемся

$S \Rightarrow aB \Rightarrow accd$ — не подходит, откатываемся

Bruteforce parsing: пример

$$\begin{array}{rcccl} S & \rightarrow & aAd & \mid & aB \\ A & \rightarrow & b & \mid & c \\ B & \rightarrow & ccd & \mid & ddc \end{array}$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd \Rightarrow acd$ — не подходит, откатываемся

$S \Rightarrow aB \Rightarrow accd$ — не подходит, откатываемся

$S \Rightarrow aB$

Bruteforce parsing: пример

$$\begin{array}{lcccc} S & \to & aAd & \mid & aB \\ A & \to & b & \mid & c \\ B & \to & ccd & \mid & ddc \end{array}$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd \Rightarrow acd$ — не подходит, откатываемся

$S \Rightarrow aB \Rightarrow accd$ — не подходит, откатываемся

$S \Rightarrow aB \Rightarrow addc$

Bruteforce parsing: пример

$$
\begin{array}{rcccl}
S & \to & aAd & | & aB \\
A & \to & b & | & c \\
B & \to & ccd & | & ddc
\end{array}
$$

$\omega = addc$

$S \Rightarrow aAd \Rightarrow abd$ — не подходит, откатываемся

$S \Rightarrow aAd \Rightarrow acd$ — не подходит, откатываемся

$S \Rightarrow aB \Rightarrow accd$ — не подходит, откатываемся

$S \Rightarrow aB \Rightarrow addc$ — ура!

Проблема: ну очень уж долго работает: экспоненциальное время!

Нисходящий синтаксический анализ без отката

- Рекурсивный спуск (recursive descent parsing)
  - Для каждого нетерминала написана функция
  - Функции для нетерминалов рекурсивно вызывают друг друга

$$S \to (S) \mid \varepsilon$$

```
parse_S word =
  if (word == empty) then (true, word)
  else
    let (r, w') = parse_lbr word in
      if (not r)
      then (false, word)
      else
        let (r, w'') = parse_S w' in
          if (not r)
          then (false, w')
          else parser_rbr w''
```

Нисходящий синтаксический анализ без отката: LL(k)

- Идея: откат запрещен, но разрешен предпросмотр
- По (нескольким) следующим терминалам принять решение о том, какую продукцию использовать
- Как и предыдущие 2 подхода не может обрабатывать леворекурсивные правила грамматики
- Достаточно хорош для используемых на практике языков

Леворекурсивные правила грамматики

- Явная (непосредственная) левая рекурсия
  - $A \to A\beta$
- Неявная левая рекурсия
  - $A \to \alpha A\beta$, $\alpha \overset{*}{\Rightarrow} \varepsilon$
- Взаимная рекурсия
  - $A \to \alpha B\beta$, $B \to \gamma A\delta$, $\alpha \overset{*}{\Rightarrow} \varepsilon, \gamma \overset{*}{\Rightarrow} \varepsilon$

Избавление от левой рекурсии

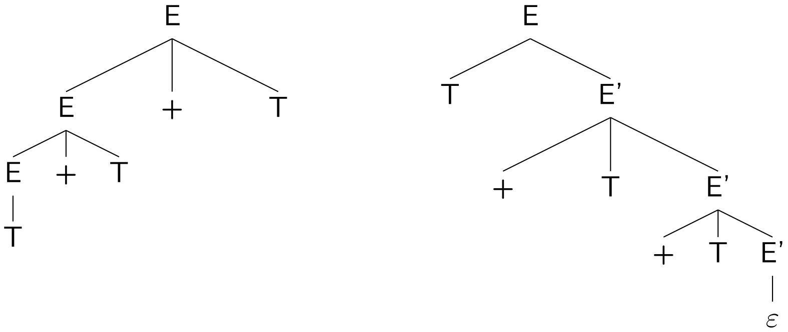
- $A \to A\alpha \mid \beta \Leftrightarrow A \to \beta A',\ A' \to \varepsilon \mid \alpha A'$

Избавление от левой рекурсии

- $A \to A\alpha \mid \beta \Leftrightarrow A \to \beta A',\ A' \to \varepsilon \mid \alpha A'$
- $E \to E + T \mid T \Leftrightarrow E \to TE',\ E' \to \varepsilon \mid +TE'$

Избавление от левой рекурсии

- $A \to A\alpha \mid \beta \Leftrightarrow A \to \beta A',\ A' \to \varepsilon \mid \alpha A'$
- $E \to E + T \mid T \Leftrightarrow E \to TE',\ E' \to \varepsilon \mid + TE'$



Избавление от левой рекурсии: более общий случай

- $A \to A\alpha_1 \,|\, A\alpha_2 \,|\, \ldots \,|\, A\alpha_n \,|\, \beta_1 \,|\, \beta_2 \,|\, \ldots \,|\, \beta_k$

- $A \to \beta_1 A' \,|\, \beta_2 A' \,|\, \ldots \,|\, \beta_k A'$
- $A' \to \varepsilon \,|\, \alpha_1 A' \,|\, \alpha_2 A' \,|\, \ldots \,|\, \alpha_n A'$

Избавление от взаимной левой рекурсии

- Избавляемся от ε-продукций
- Упорядочиваем правила по индексу нетерминала
- Добиваемся того, чтобы не было правил вида $A_i \to A_j\alpha, j \leq i$
  - Перебираем все A_i
  - Перебираем все $A_j, 1 \leq j < i$
  - Для каждого правила $p : A_i \to A_j\gamma$
    - Удалить правило p
    - Для каждого правила $A_j \to \delta_1 \mid \ldots \mid \delta_k$ Добавить правила $A_i \to \delta_l$
  - Устранить непосредственную левую рекурсию для A_i

Левая факторизация грамматики

- Выделяем наибольший общий префикс продукций
  $A \to \alpha\beta \,|\, \alpha\gamma \Rightarrow A \to \alpha A',\ A' \to \beta \,|\, \gamma$

Пример

$$
\begin{aligned}
S &\rightarrow aSSbS \\
&\mid aSaSb \\
&\mid abb \\
&\mid b
\end{aligned}
$$

Пример

$$
\begin{aligned}
S &\rightarrow aSSbS \\
&\mid aSaSb \\
&\mid abb \\
&\mid b
\end{aligned}
$$

$$
\begin{aligned}
S &\rightarrow aS' \\
&\mid b \\
S' &\rightarrow SSbS \\
&\mid SaSb \\
&\mid bb
\end{aligned}
$$

Пример

$$\begin{array}{rcl} S & \to & aSSbS \\ & | & aSaSb \\ & | & abb \\ & | & b \end{array}$$

$$\begin{array}{rcl} S & \to & aS' \\ & | & b \\ S' & \to & SSbS \\ & | & SaSb \\ & | & bb \end{array}$$

$$\begin{array}{rcl} S & \to & aS' \mid b \\ S' & \to & SS'' \mid bb \\ S'' & \to & SbS \mid aSb \end{array}$$

Множество FIRST

- Множество символов, которые могут появиться первыми во время вывода из данной сентенциальной формы
- $FIRST(a\alpha) = \{a\}$, $a \in V_T, \alpha \in (V_T \cup V_N)^*$
- $FIRST(\varepsilon) = \{\varepsilon\}$
- $FIRST(\alpha\beta) = FIRST(\alpha) \cup (FIRST(\beta), if \, \varepsilon \in FIRST(\alpha))$
- $FIRST(S) = FIRST(\alpha) \cup FIRST(\beta), S \to \alpha \,|\, \beta$

Множество FIRST: пример

$$
\begin{array}{rcl}
S & \to & aS' \\
S' & \to & AbBS' \mid \varepsilon \\
A & \to & aA' \mid \varepsilon \\
A' & \to & b \mid a \\
B & \to & c \mid \varepsilon
\end{array}
$$

Множество FIRST: пример

$$
\begin{array}{rcl}
S & \rightarrow & aS' \\
S' & \rightarrow & AbBS' \mid \varepsilon \\
A & \rightarrow & aA' \mid \varepsilon \\
A' & \rightarrow & b \mid a \\
B & \rightarrow & c \mid \varepsilon
\end{array}
$$

- $FIRST(S) = \{a\}$

Множество FIRST: пример

$$
\begin{array}{rcl}
S & \to & aS' \\
S' & \to & AbBS' \mid \varepsilon \\
A & \to & aA' \mid \varepsilon \\
A' & \to & b \mid a \\
B & \to & c \mid \varepsilon
\end{array}
$$

- $FIRST(S) = \{a\}$
- $FIRST(A) = \{a, \varepsilon\}$

Множество FIRST: пример

$$
\begin{array}{rcl}
S & \to & aS' \\
S' & \to & AbBS' \mid \varepsilon \\
A & \to & aA' \mid \varepsilon \\
A' & \to & b \mid a \\
B & \to & c \mid \varepsilon
\end{array}
$$

- $FIRST(S) = \{a\}$
- $FIRST(A) = \{a, \varepsilon\}$
- $FIRST(A') = \{a, b\}$

Множество FIRST: пример

$$\begin{array}{rcl} S & \to & aS' \\ S' & \to & AbBS' \mid \varepsilon \\ A & \to & aA' \mid \varepsilon \\ A' & \to & b \mid a \\ B & \to & c \mid \varepsilon \end{array}$$

- $FIRST(S) = \{a\}$
- $FIRST(A) = \{a, \varepsilon\}$
- $FIRST(A') = \{a, b\}$
- $FIRST(B) = \{c, \varepsilon\}$

Множество FIRST: пример

$$
\begin{array}{rcl}
S & \rightarrow & aS' \\
S' & \rightarrow & AbBS' \mid \varepsilon \\
A & \rightarrow & aA' \mid \varepsilon \\
A' & \rightarrow & b \mid a \\
B & \rightarrow & c \mid \varepsilon
\end{array}
$$

- $FIRST(S) = \{a\}$
- $FIRST(A) = \{a, \varepsilon\}$
- $FIRST(A') = \{a, b\}$
- $FIRST(B) = \{c, \varepsilon\}$
- $FIRST(S') = \{a, b, \varepsilon\}$

Множество FOLLOW

- Множество символов, которые могут появиться в некотором выводе сразу после данной сентенциальной формы
- Положим $FOLLOW(X) = \varnothing$
- Если X — стартовый нетерминал, $FOLLOW(X) = FOLLOW(X) \cup \{\$\}$ — символ конца строки
- Для всех правил вида $A \to \alpha X \beta$, $FOLLOW(X) = FOLLOW(X) \cup (FIRST(\beta) \setminus \{\varepsilon\})$
- Для всех правил вида $A \to \alpha X$ и $A \to \alpha X \beta$, где $\varepsilon \in FIRST(\beta)$, $FOLLOW(X) = FOLLOW(X) \cup FOLLOW(A)$
- Повторять последние 2 пункта, пока можно что-то добавлять

Множество FOLLOW: пример

$$
\begin{array}{rcl}
S & \to & aS' \\
S' & \to & AbBS' \mid \varepsilon \\
A & \to & aA' \mid \varepsilon \\
A' & \to & b \mid a \\
B & \to & c \mid \varepsilon
\end{array}
$$

Множество FOLLOW: пример

$$
\begin{array}{rcl}
S & \rightarrow & aS' \\
S' & \rightarrow & AbBS' \mid \varepsilon \\
A & \rightarrow & aA' \mid \varepsilon \\
A' & \rightarrow & b \mid a \\
B & \rightarrow & c \mid \varepsilon
\end{array}
$$

- $FOLLOW(S) = \{\$\}$

Множество FOLLOW: пример

$$
\begin{array}{rcl}
S & \to & aS' \\
S' & \to & AbBS' \mid \varepsilon \\
A & \to & aA' \mid \varepsilon \\
A' & \to & b \mid a \\
B & \to & c \mid \varepsilon
\end{array}
$$

- $FOLLOW(S) = \{\$\}$
- $FOLLOW(S') = \{\$\} \, (S \to aS')$

Множество FOLLOW: пример

$$\begin{array}{rcl} S & \to & aS' \\ S' & \to & AbBS' \mid \varepsilon \\ A & \to & aA' \mid \varepsilon \\ A' & \to & b \mid a \\ B & \to & c \mid \varepsilon \end{array}$$

- $FOLLOW(S) = \{\$\}$
- $FOLLOW(S') = \{\$\}\,(S \to aS')$
- $FOLLOW(A) = \{b\}\,(S' \to AbBS')$

Множество FOLLOW: пример

$$
\begin{array}{rcl}
S & \to & aS' \\
S' & \to & AbBS' \mid \varepsilon \\
A & \to & aA' \mid \varepsilon \\
A' & \to & b \mid a \\
B & \to & c \mid \varepsilon
\end{array}
$$

- $FOLLOW(S) = \{\$\}$
- $FOLLOW(S') = \{\$\}\,(S \to aS')$
- $FOLLOW(A) = \{b\}\,(S' \to AbBS')$
- $FOLLOW(A') = \{b\}\,(A \to aA')$

Множество FOLLOW: пример

$$
\begin{array}{rcl}
S & \to & aS' \\
S' & \to & AbBS' \mid \varepsilon \\
A & \to & aA' \mid \varepsilon \\
A' & \to & b \mid a \\
B & \to & c \mid \varepsilon
\end{array}
$$

- $FOLLOW(S) = \{\$\}$
- $FOLLOW(S') = \{\$\}\,(S \to aS')$
- $FOLLOW(A) = \{b\}\,(S' \to AbBS')$
- $FOLLOW(A') = \{b\}\,(A \to aA')$
- $FOLLOW(B) = \{a, b, \$\}\,(S' \to AbBS', \varepsilon \in FIRST(S'))$

LL(1)-анализ

- Нисходящий синтаксический анализ с предпросмотром одного символа
- Читает вход слева направо (L: left-to-right), строит левый вывод в грамматике (L: leftmost)
- Состоит из:
  - Входного буфера (откуда читается входная строка)
  - Стека (для промежуточных данных)
  - Таблицы анализатора (управляет процессом разбора)
- Работает за $O(n)$, где n — длина входной строки

Таблица LL(1)-анализатора

$$S \to (S) \mid \varepsilon$$

Размещаем продукции в таблице (по горизонтали — нетерминалы; по вертикали — терминалы + \$)

- Продукции вида $A \to \alpha$ — в ячейки (A, a), где $a \in FIRST(A)$
- Продукции вида $A \to \varepsilon$ — в ячейки (A, a), где $a \in FOLLOW(A)$

N	FIRST	FOLLOW	(	)	\$
S					

Таблица LL(1)-анализатора

$$S \to (S) \mid \varepsilon$$

Размещаем продукции в таблице (по горизонтали — нетерминалы; по вертикали — терминалы + \$)

- Продукции вида $A \to \alpha$ — в ячейки (A, a), где $a \in FIRST(A)$
- Продукции вида $A \to \varepsilon$ — в ячейки (A, a), где $a \in FOLLOW(A)$

N	FIRST	FOLLOW	(	)	\$
S	$\{(, \varepsilon\}$	$\{), \$\}$			

Таблица LL(1)-анализатора

$$S \to (S) \mid \varepsilon$$

Размещаем продукции в таблице (по горизонтали — нетерминалы; по вертикали — терминалы + $)

- Продукции вида $A \to \alpha$ — в ячейки (A, a), где $a \in FIRST(A)$
- Продукции вида $A \to \varepsilon$ — в ячейки (A, a), где $a \in FOLLOW(A)$

N	FIRST	FOLLOW	(	)	$
S	$\{(, \varepsilon\}$	$\{), \$\}$	$S \to (S)$		

Таблица LL(1)-анализатора

$$S \to (S) \mid \varepsilon$$

Размещаем продукции в таблице (по горизонтали — нетерминалы; по вертикали — терминалы + \$)

- Продукции вида $A \to \alpha$ — в ячейки (A, a), где $a \in FIRST(A)$
- Продукции вида $A \to \varepsilon$ — в ячейки (A, a), где $a \in FOLLOW(A)$

N	FIRST	FOLLOW	(	)	\$
S	$\{(, \varepsilon\}$	$\{), \$\}$	$S \to (S)$	$S \to \varepsilon$	$S \to \varepsilon$

Синтаксический анализ (доска)

$$S \to (S) \mid \varepsilon$$

N	FIRST	FOLLOW	(	)	\$
S	$\{(,\varepsilon\}$	$\{),\$\}$	$S \to (S)$	$S \to \varepsilon$	$S \to \varepsilon$

$\omega = (())\$$
Стек: $\$, S,), S, (,), S, ($

Когда LL-анализ не возможен

- Леворекурсивные правила
- Когда при построении таблицы в одну ячейку нужно записать больше одной записи
  - FIRST-FIRST конфликт
    - $A \to \alpha \mid \beta, FIRST(\alpha) \cap FIRST(\beta) \neq \varnothing$
    - $E \to T + E \mid T * E$
  - FIRST-FOLLOW конфликт
    - $FIRST(A) \cap FOLLOW(A) \neq \varnothing$
    - $S \to Aab, A \to a \mid \varepsilon$
- Как с этим бороться?
  - Избавиться от левой рекурсии
  - Избавиться от недетерминизма
  - Факторизовать грамматику
  - Использовать аннотации (если есть)
  - Переписать грамматику
  - Использовать более одного символа предпросмотра

Нисходящий синтаксический анализ: функция *FIRST*

- Функция $FIRST_k^G(\alpha) = \{\omega \in V_T^* \mid$ либо $|\omega| < k$ и $\alpha \stackrel{*}{\Rightarrow} \omega$, либо $|\omega| = k$ и $\alpha \stackrel{*}{\Rightarrow} \omega\gamma, \gamma \in V_T^*\}$
  - По сути: первые k символов, встречающиеся в выводе из α
- Пример
  - $S \to SS \mid aSb \mid \varepsilon$
  - $FIRST_3^G(aSb) = \{ab, aab, aaa\}$
  - $aba \notin FIRST_3^G(aSb)$!

Нисходящий синтаксический анализ: LL-грамматики

Фундаментальное свойство: по сентенциальной форме $a_1 a_2 \ldots a_j A\beta, a_i \in V_T, A \in V_N, \beta \in (V_T \cup V_N)^*$ однозначно определяется, какое правило нужно применять дальше, чтобы разобрать всю строку

Нисходящий синтаксический анализ: LL-грамматики

Фундаментальное свойство: по сентенциальной форме $a_1 a_2 \ldots a_j A\beta, a_i \in V_T, A \in V_N, \beta \in (V_T \cup V_N)^*$ однозначно определяется, какое правило нужно применять дальше, чтобы разобрать всю строку

КС грамматика G является **LL(k)-грамматикой** для некоторого k, если для любых двух левосторонних выводов вида

- $S \overset{*}{\Rightarrow} \omega A\alpha \Rightarrow \omega\beta\alpha \overset{*}{\Rightarrow} \omega\delta$
- $S \overset{*}{\Rightarrow} \omega A\alpha \Rightarrow \omega\gamma\alpha \overset{*}{\Rightarrow} \omega\eta$

в которых $FIRST_k^G(\delta) = FIRST_k^G(\eta)$, верно $\beta = \gamma$

КС грамматика G является **LL-грамматикой**, если она является $LL(k)$-грамматикой для некоторого $k \geq 0$

Пример LL(1)-грамматики

$S \to aBS \mid b$
$B \to a \mid bSB$
Надо показать: для любых левосторонних выводов

- $S \overset{*}{\Rightarrow} \omega A\alpha \Rightarrow \omega\beta\alpha \overset{*}{\Rightarrow} \omega\delta$
- $S \overset{*}{\Rightarrow} \omega A\alpha \Rightarrow \omega\gamma\alpha \overset{*}{\Rightarrow} \omega\eta$

если δ и η начинаются с одного символа, то $\beta = \gamma$
Рассматриваем выводы, где роль A выполняет S: $S \Rightarrow aBS, S \Rightarrow b$.
$\omega = \alpha = \varepsilon, \beta = aBS, \gamma = b$. Любая цепочка, выводимая из $\beta\alpha = aBS$ начинается на a; любая цепочка, выводимая из $\gamma\alpha = b$ начинается на b. Однозначно определяется, какой альтернативе следовать.
Аналогично с $A = B : S \Rightarrow aBS \Rightarrow aaS; S \Rightarrow aBS \Rightarrow abSBS$

Простая LL(1)-грамматика

КС-грамматика G называется **простой LL(1)-грамматикой**, если в ней нет ε-правил, и все альтернативы для каждого нертерминала начинаются с терминалов, и притом различных.

$\forall (A, a), A \in V_N, a \in V_T, \exists$ самое большое 1 альтернатива вида $A \rightarrow a\alpha$

LL-грамматика: необходимое и достаточное условие

Теорема

КС грамматика $G = \langle V_N, V_T, P, S \rangle$ *является* $LL(k)$*-грамматикой*
$\Leftrightarrow FIRST_k^G(\beta\alpha) \cap FIRST_k^G(\gamma\alpha) = \varnothing$, *для всех таких*
$\alpha, \beta, \gamma : A \to \beta, A \to \gamma \in P, \beta \neq \gamma, \exists$ *вывод* $S \overset{*}{\Rightarrow} \omega A\alpha$

LL-грамматика: функция FOLLOW

$$FOLLOW_k^G(\beta) = \{\omega \in V_T^* \mid S \stackrel{*}{\Rightarrow} \gamma\beta\alpha, \omega \in FIRST_k^G(\alpha)\}, k \geq 0$$

Пример: $S \to SS \mid aSb \mid \varepsilon$

- $FOLLOW_3^G(aa) = \{abb, aab, aaa, aba, baa, bab, bb, bba, \dots\}$
- $\varepsilon, b \notin FOLLOW_3^G$!

LL(1)-грамматика: необходимое и достаточное условие

Теорема

КС-грамматика $G = \langle V_N, V_T, P, S\rangle$ *является* $LL(1)$*-грамматикой* $\Leftrightarrow FIRST_1^G(\beta FOLLOW_1^G(A)) \cap FIRST_1^G(\gamma FOLLOW_1^G(A)) = \varnothing, \forall A \in V_N, \beta, \gamma \in (V_N \cup V_T)^*, A \to \gamma, A \to \beta \in P, \beta \neq \gamma$

LL(1)-грамматика: необходимое и достаточное условие: другая формулировка

Теорема

КС-грамматика $G = \langle V_N, V_T, P, S \rangle$ *является* $LL(1)$*-грамматикой* $\Leftrightarrow \forall A \to \alpha_1 \,|\, \alpha_2 \,|\, \ldots \,|\, \alpha_n$ *верно:*

- $FIRST_1^G(\alpha_i) \cap FIRST_1^G(\alpha_j) = \varnothing, i \neq j, 1 \leq i, j \leq n$
- *если* $\alpha_i \overset{*}{\Rightarrow} \varepsilon$, *то* $FIRST_1^G(\alpha_j) \cap FOLLOW_1^G(A) = \varnothing, 1 \leq j \leq n, i \neq j$

Леворекурсивность

Теорема

Если КС-грамматика $G = \langle V_N, V_T, P, S\rangle$ леворекурсивна, то она не является $LL(k)$-грамматикой ни при каком k