

Теория автоматов и формальных языков

Контекстно-свободные языки: нисходящий анализ

Лектор: Екатерина Вербицкая

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

07 ноября 2019

- Контекстно-свободные грамматики (все правила вида $A \rightarrow \alpha$)
- КС языки и разрешимость проверки пустоты
- Нормальная форма Хомского
- Алгоритм СЮК

В предыдущей серии: НФХ

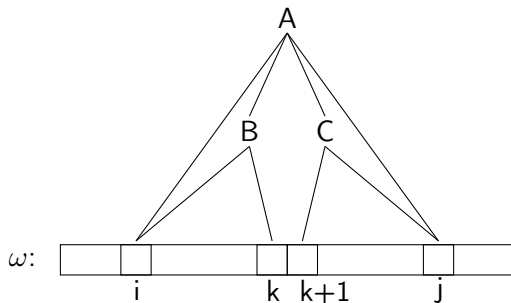
КС грамматика находится в **нормальной форме Хомского**, если все ее правила имеют вид:

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

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

В предыдущей серии: СΥΚ

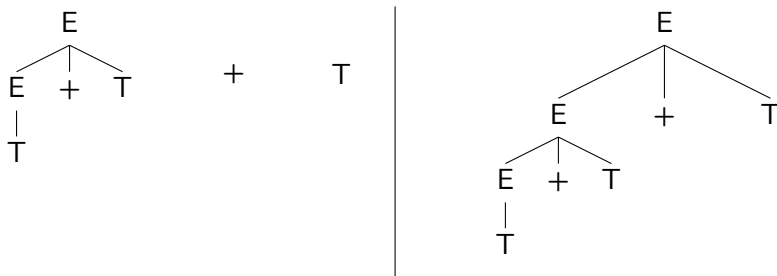
- Алгоритм синтаксического анализа, работающий с грамматиками в НФХ
- Динамическое программирование



- Дано: строка ω длины n , грамматика $G = \langle V_T, V_N, P, S \rangle$ в НФХ
- Используем трехмерный массив d булевых значений размером $|V_N| \times n \times n$, $d[A][i][j] = \text{true} \Leftrightarrow A \Rightarrow \omega[i \dots j]$
- 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 $\{(i', j') \mid j' - i' < m\}$
 - ▶ $d[A][i][j] = \bigvee_{A \rightarrow BC} \bigvee_{k=i}^{j-1} d[B][i][k] \wedge d[C][k][j]$
- В конце работы алгоритма в $d[S][0][n]$ записан ответ, выводится ли ω в данной грамматике

СНК — алгоритм восходящего анализа

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



Восходящий анализ контринтуитивен
(особенно при диагностике ошибок)

- 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

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

$$\omega = addc$$

Bruteforce parsing: пример

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

$$\omega = addc$$

S

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd$$

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd$$

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd \Rightarrow acd$$

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd \Rightarrow acd$$

не подходит, откатываемся

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd \Rightarrow acd$$

S

не подходит, откатываемся

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd \Rightarrow acd$$

$$S \Rightarrow aB$$

не подходит, откатываемся

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd \Rightarrow acd$$

$$S \Rightarrow aB \Rightarrow accd$$

не подходит, откатываемся

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd \Rightarrow acd$$

$$S \Rightarrow aB \Rightarrow accd$$

не подходит, откатываемся

не подходит, откатываемся

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd \Rightarrow acd$$

$$S \Rightarrow aB \Rightarrow accd$$

$$S \Rightarrow aB$$

не подходит, откатываемся

не подходит, откатываемся

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd \Rightarrow acd$$

$$S \Rightarrow aB \Rightarrow accd$$

$$S \Rightarrow aB \Rightarrow addc$$

не подходит, откатываемся

не подходит, откатываемся

не подходит, откатываемся

Bruteforce parsing: пример

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

$$\omega = addc$$

$$S \Rightarrow aAd \Rightarrow abd$$

$$S \Rightarrow aAd \Rightarrow acd$$

$$S \Rightarrow aB \Rightarrow accd$$

$$S \Rightarrow aB \Rightarrow addc$$

не подходит, откатываемся

не подходит, откатываемся

не подходит, откатываемся

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

$$S \rightarrow (S) \mid \varepsilon$$

```
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(1)

- Идея: откат запрещен, но разрешен предпросмотр
- По следующему терминалу 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

- 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

Управляет процессом разбора: показывает, какую продукцию применять, если во время анализа рассматривается нетерминал A , а следующий символ входа — t

	...	t	...	\$

A	...	$A \rightarrow \alpha$

Для заполнения таблицы надо научиться считать множества символов, которые можно встретить во время анализа

Множество FIRST

Множество символов, которые могут появиться первыми во время вывода из данной сентенциальной формы

- $FIRST(a\alpha) = \{a\}$, если $a \in V_T, \alpha \in (V_T \cup V_N)^*$
- $FIRST(\varepsilon) = \{\varepsilon\}$
- $FIRST(\alpha\beta) = FIRST(\alpha) \cup FIRST(\beta)$, если $\varepsilon \in FIRST(\alpha)$
- $FIRST(S) = FIRST(\alpha) \cup FIRST(\beta)$, если есть правило $S \rightarrow \alpha \mid \beta$

Множество FIRST: пример

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

Множество FIRST: пример

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

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

Множество FIRST: пример

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

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

Множество FIRST: пример

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

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

Множество FIRST: пример

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

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

Множество FIRST: пример

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

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

Множество FOLLOW

Множество символов, которые могут появиться в некотором выводе сразу после данной сентенциальной формы

- Положим $FOLLOW(X) = \emptyset$
- Если X — стартовый нетерминал,
 $FOLLOW(X) = FOLLOW(X) \cup \{\$$ — символ конца строки
- Для всех правил вида $A \rightarrow \alpha X \beta$,
 $FOLLOW(X) = FOLLOW(X) \cup (FIRST(\beta) \setminus \{\epsilon\})$
- Для всех правил вида $A \rightarrow \alpha X$ и $A \rightarrow \alpha X \beta$, где $\epsilon \in FIRST(\beta)$,
 $FOLLOW(X) = FOLLOW(X) \cup FOLLOW(A)$
- Повторять последние 2 пункта, пока можно что-то добавлять

Множество FOLLOW: пример

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

Множество FOLLOW: пример

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

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

Множество FOLLOW: пример

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

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

$$(S \rightarrow aS')$$

Множество FOLLOW: пример

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

- $FOLLOW(S) = \{\$ \}$
- $FOLLOW(S') = \{\$ \}$
- $FOLLOW(A) = \{b\}$

$$\begin{aligned} &(S \rightarrow aS') \\ &(S' \rightarrow AbBS') \end{aligned}$$

Множество FOLLOW: пример

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

- $FOLLOW(S) = \{\$ \}$
- $FOLLOW(S') = \{\$ \}$
- $FOLLOW(A) = \{b\}$
- $FOLLOW(A') = \{b\}$

$$(S \rightarrow aS')$$

$$(S' \rightarrow AbBS')$$

$$(A \rightarrow aA')$$

Множество FOLLOW: пример

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

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

Таблица LL(1)-анализатора

$$S \rightarrow (S) \mid \varepsilon$$

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

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

N	FIRST	FOLLOW	()	\$
S					

Таблица LL(1)-анализатора

$$S \rightarrow (S) \mid \varepsilon$$

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

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

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

Таблица LL(1)-анализатора

$$S \rightarrow (S) \mid \varepsilon$$

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

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

N	FIRST	FOLLOW	()	\$
S	{(, ε }	{), \$}	$S \rightarrow (S)$		

Таблица LL(1)-анализатора

$$S \rightarrow (S) \mid \varepsilon$$

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

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

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

LL(1)-анализ

- Инициализация: указатель в строке на первый символ, в стек помещаем $\$$ и стартовый нетерминал
- Пока стек не пуст
 - ▶ Если на вершине стека нетерминал N , указатель в строке на символе t , смотрим на содержимое ячейки (N, t) управляющей таблицы
 - ★ Если ячейка пуста, сообщаем об ошибке анализа
 - ★ Если в ячейке продукция $N \rightarrow \varepsilon$, снимаем со стека N
 - ★ Если в ячейке продукция $N \rightarrow \alpha$, снимаем со стека N , символы α кладем на стек в обратном порядке
 - ▶ Если на вершине стека терминал t
 - ★ Если указатель в строке на терминале t , снимаем со стека вершину, двигаем указатель на следующий символ
 - ★ Если указатель в строке на любом другом терминале, сообщаем об ошибке
- Если строка прочитана полностью, анализ завершен успешно. Иначе — полагается сообщить об ошибке

Пример (доска)

$$S \rightarrow (S) \mid \varepsilon$$

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

$$\omega = (())\$$$

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
 - ★ $A \rightarrow \alpha \mid \beta, FIRST(\alpha) \cap FIRST(\beta) \neq \emptyset$
 - ★ $E \rightarrow T + E \mid T * E$
 - ▶ FIRST-FOLLOW конфликт
 - ★ $FIRST(A) \cap FOLLOW(A) \neq \emptyset$
 - ★ $S \rightarrow Aab, A \rightarrow a \mid \varepsilon$
- Как с этим бороться?
 - ▶ Избавиться от левой рекурсии
 - ▶ Избавиться от недетерминизма
 - ▶ Факторизовать грамматику
 - ▶ Использовать аннотации (если есть)
 - ▶ Переписать грамматику
 - ▶ Использовать более одного символа предпросмотра

Леворекурсивные правила грамматики

- Явная (непосредственная) левая рекурсия
 - ▶ $A \rightarrow A\beta$
- Неявная левая рекурсия
 - ▶ $A \rightarrow \alpha A\beta, \alpha \xRightarrow{*} \varepsilon$
- Взаимная рекурсия
 - ▶ $A \rightarrow \alpha B\beta, B \rightarrow \gamma A\delta, \alpha \xRightarrow{*} \varepsilon, \gamma \xRightarrow{*} \varepsilon$

Избавление от левой рекурсии

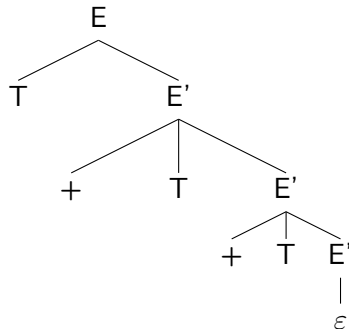
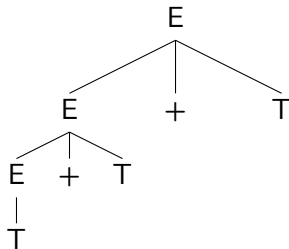
- $A \rightarrow A\alpha \mid \beta \Leftrightarrow A \rightarrow \beta A', A' \rightarrow \varepsilon \mid \alpha A'$

Избавление от левой рекурсии

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

Избавление от левой рекурсии

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



Избавление от левой рекурсии: более общий случай

- $A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \dots \mid \beta_k$
- $A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \dots \mid \beta_k A'$
- $A' \rightarrow \varepsilon \mid \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_n A'$

Избавление от взаимной левой рекурсии

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

Левая факторизация грамматики

- Выделяем наибольший общий префикс продукций
 $A \rightarrow \alpha\beta \mid \alpha\gamma \Rightarrow A \rightarrow \alpha A', A' \rightarrow \beta \mid \gamma$

Пример

$$\begin{array}{lcl} S & \rightarrow & aSSbS \\ & | & aSaSb \\ & | & abb \\ & | & b \end{array}$$

Пример

$$\begin{array}{lcl} S & \rightarrow & aSSbS \\ & | & aSaSb \\ & | & abb \\ & | & b \end{array}$$

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

Пример

$$\begin{array}{lcl} S & \rightarrow & aSSbS \\ & | & aSaSb \\ & | & abb \\ & | & b \end{array}$$

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

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

- Можно использовать более одного символа предпросмотра
- Все равно применимо не ко всем КС-грамматикам

LL(k): функция *FIRST*

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

LL(k): функция FOLLOW

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

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

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

Нисходящий синтаксический анализ: LL-грамматики

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

Нисходящий синтаксический анализ: LL-грамматики

Фундаментальное свойство: по сентенциальной форме $a_1 a_2 \dots a_j A \beta$, $a_i \in V_T$, $A \in V_N$, $\beta \in (V_T \cup V_N)^*$ 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

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

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

$$S \rightarrow aBS \mid b$$

$$B \rightarrow a \mid bSB$$

Надо показать: для любых левосторонних выводов

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

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.

$\forall (A, a), A \in V_N, a \in V_T, \exists$ максимум 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
 $\Leftrightarrow FIRST_k^G(\beta\alpha) \cap FIRST_k^G(\gamma\alpha) = \emptyset$, для всех таких
 $\alpha, \beta, \gamma : A \rightarrow \beta, A \rightarrow \gamma \in P, \beta \neq \gamma, \exists$ вывод $S \xRightarrow{*} \omega A \alpha$

LL(1)-грамматика: необходимое и достаточное условие

Теорема

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

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

$\Leftrightarrow \forall A \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$ верно:

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

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