

Нисходящий разбор



Теория формальных языков
2021 г.



Неоднозначные КС-языки

Рассмотрим КС-язык $\{a^n b^m c^m\} \cup \{a^n b^n c^m\}$. Слова $a^n b^n c^n$ этого языка гарантированно имеют минимум два дерева разбора.

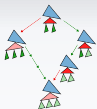


Неоднозначные КС-языки

Рассмотрим КС-язык $\{a^n b^m c^m\} \cup \{a^n b^n c^m\}$. Слова $a^n b^n c^n$ этого языка гарантированно имеют минимум два дерева разбора.

Определение

КС-грамматика G неоднозначная, если существует слово $w \in L(G)$ такое, что в G у него больше одного дерева разбора. КС-язык L существенно неоднозначен, если всякая его грамматика неоднозначна.



Неоднозначные КС-языки

Рассмотрим КС-язык $\{a^n b^m c^m\} \cup \{a^n b^n c^m\}$. Слова $a^n b^n c^n$ этого языка гарантированно имеют минимум два дерева разбора.

Определение

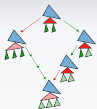
КС-грамматика G неоднозначная, если существует слово $w \in L(G)$ такое, что в G у него больше одного дерева разбора. КС-язык L существенно неоднозначен, если всякая его грамматика неоднозначна.

Существование однозначной грамматики не гарантирует существования DPDA: см. $\{a^n b^n\} \cup \{a^n b^{2n}\}$.



Алгоритм Касами–Тории

Алгоритм Кока–Янгера–Касами — таблица
 $T_{i,j} = \{A \mid a_{i+1} \dots a_j \in L_G(A)\}$; изменим её на
 $T'_j[A] = \{i \mid a_{i+1} \dots a_j \in L_G(A)\}.$

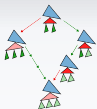


Алгоритм Касами–Тории

Алгоритм Кока–Янгера–Касами — таблица
 $T_{i,j} = \{A \mid a_{i+1} \dots a_j \in L_G(A)\}$; изменим её на
 $T'_j[A] = \{i \mid a_{i+1} \dots a_j \in L_G(A)\}$.

Идея алгоритма СТ

Читаем очередной символ, вычисляем множество $T'_j[A]$ для всех $A \in N$, постепенно достраивая его как список, упорядоченный по возрастанию. Если $A \rightarrow BC$, тогда если $k \in T'_j[C]$, то для всех $x \in T'_k[B]$ выполнено $x \in T'_j[A]$.

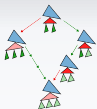


Алгоритм Касами–Тории

Алгоритм Кока–Янгера–Касами — таблица
 $T_{i,j} = \{A \mid a_{i+1} \dots a_j \in L_G(A)\}$; изменим её на
 $T'_j[A] = \{i \mid a_{i+1} \dots a_j \in L_G(A)\}$.

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for  $j=1..n$ 
  for all  $A \in N$  if  $A \rightarrow a_j \in R$  then  $T'_j[A] = \{j-1\}$ 
  for  $k=j-1..1$ 
    for all  $A \rightarrow BC \in R$ 
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
```



Алгоритм Касами–Тории

Алгоритм Кока–Янгера–Касами — таблица
 $T_{i,j} = \{A \mid a_{i+1} \dots a_j \in L_G(A)\}$; изменим её на
 $T'_j[A] = \{i \mid a_{i+1} \dots a_j \in L_G(A)\}$.

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for  $j=1..n$ 
  for all  $A \in N$  if  $A \rightarrow a_j \in R$  then  $T'_j[A] = \{j-1\}$ 
  for  $k=j-1..1$ 
    for all  $A \rightarrow BC \in R$ 
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 

```

Для однозначных грамматик АСТ работает за $O(n^2)$.

$S \rightarrow G_A A \mid G_B B \mid a \mid b$ $A \rightarrow a \mid S G_A$ $B \rightarrow b \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow b$ слово abba

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for j=1..n
  for k=j-1..1
    for all  $A \rightarrow BC \in R$ 
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
    
```

Инициализация таб-
лицы:

	S	A	B	G_A	G_B
1 — a					
2 — b					
3 — b					
4 — a					

$S \rightarrow G_A A \mid G_B B \mid a \mid b$ $A \rightarrow a \mid S G_A$ $B \rightarrow b \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow b$ слово abba

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for  $j=1..n$  \\  $j = 1$ 
  for all  $A \in N$  if  $A \rightarrow a_j \in R$  then  $T'_j[A] = \{j-1\}$ 
  for  $k=j-1..1$ 
    for all  $A \rightarrow BC \in R$ 
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
    
```

$j = 1$, проход по
 терминальным прави-
 лам, цикла по нетер-
 минальным нет, т.к.
 $k = 0$:

	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b					
3 — b					
4 — a					

$S \rightarrow G_A A \mid G_B B \mid a \mid \mathbf{b}$ $A \rightarrow a \mid S G_A$ $B \rightarrow \mathbf{b} \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow \mathbf{b}$ слово $abba$

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for  $j=1..n$   $\backslash \backslash j=2$ 
  for all  $A \in N$  if  $A \rightarrow a_j \in R$  then  $T'_j[A] = \{j-1\}$ 
  for  $k=j-1..1$ 
    for all  $A \rightarrow BC \in R$ 
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 

```

$j = 2$, проход по
 терминальным прави-
 лам:

	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b	1		1		1
3 — b					
4 — a					

$S \rightarrow G_A A \mid G_B B \mid a \mid b$ $A \rightarrow a \mid S G_A$ $B \rightarrow b \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow b$ слово $abba$

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for j=1..n  \ \ j=2
  for k=j-1..1  \ \ k=1
    for all  $A \rightarrow BC \in R$   \ \  если второй нетерминал
      правой части есть в строке 2 с индексом 1
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
      \ \ тогда добавляем в ячейку второй строки для
      левого нетерминала содержимое ячейки первого
      нетерминала в строке 1

```

Подходящие правила есть для B и G_B . Но для нетерминала G_B (правило $S \rightarrow G_B B$) ячейка в первой строке пуста, так что добавляем только содержимое ячейки для S в ячейку B .

	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b	1		1, 0		1
3 — b					
4 — a					

$S \rightarrow G_A A \mid G_B B \mid a \mid \mathbf{b}$ $A \rightarrow a \mid S G_A$ $B \rightarrow \mathbf{b} \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow \mathbf{b}$ слово $abba$

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for  $j=1..n$   $\backslash \backslash j=3$ 
  for all  $A \in N$  if  $A \rightarrow a_j \in R$  then  $T'_j[A] = \{j-1\}$ 
  for  $k=j-1..1$ 
    for all  $A \rightarrow BC \in R$ 
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 

```

$j = 3$, проход по
 терминальным прави-
 лам:

	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b	1		1, 0		1
3 — b	2		2		2
4 — a					

$S \rightarrow G_A A \mid G_B B \mid a \mid b$ $A \rightarrow a \mid S G_A$ $B \rightarrow b \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow b$ слово abba

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for j=1..n  \ \ j=3
  for k=j-1..1  \ \ k=2
    for all  $A \rightarrow BC \in R$   \ \  если второй нетерминал
      правой части есть в строке 3 с индексом 2
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
      \ \ тогда добавляем в ячейку третьей строки для
      левого нетерминала содержимое ячейки первого
      нетерминала в строке 2
    
```

Подходящие правила:
 $S \rightarrow G_B B$, $B \rightarrow S G_B$,
 причём во второй строке
 ячейки G_B и S обе не
 пустые. Добавляем их
 содержимое в ячейки
 левых частей, S и B :

	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b	1		1, 0		1
3 — b	2, 1		2, 1		2
4 — a	, 0				

$S \rightarrow G_A A \mid G_B B \mid a \mid b$ $A \rightarrow a \mid S G_A$ $B \rightarrow b \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow b$ слово $abba$

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for j=1..n  \ \ j=3
  for k=j-1..1  \ \ k=1
    for all  $A \rightarrow BC \in R$   \ \  если второй нетерминал
      правой части есть в строке 3 с индексом 1
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
      \ \ тогда добавляем в ячейку третьей строки для
      левого нетерминала содержимое ячейки первого
      нетерминала в строке 1

```

1 в третьей строке есть у
 нетерминалов S и B , но
 первый никогда не быва-
 ет вторым в правой части.
 Остаётся правило $S \rightarrow$
 $G_B B$, оно также ничего не
 даёт, т.к. в первой строке
 ячейка G_B пуста.

	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b	1		1, 0		1
3 — b	2, 1		2, 1		2
4 — a	, 0				

$S \rightarrow G_A A \mid G_B B \mid \mathbf{a} \mid \mathbf{b}$ $A \rightarrow \mathbf{a} \mid S G_A$ $B \rightarrow \mathbf{b} \mid S G_B$
 $G_A \rightarrow \mathbf{a}$ $G_B \rightarrow \mathbf{b}$ слово abba

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for  $j=1..n \setminus j=4$ 
  for all  $A \in N$  if  $A \rightarrow a_j \in R$  then  $T'_j[A] = \{j-1\}$ 
  for  $k=j-1..1$ 
    for all  $A \rightarrow BC \in R$ 
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
    
```

$j = 4$, проход по
 терминальным прави-
 лам:

	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b	1		1, 0		1
3 — b	2, 1		2, 1		2
4 — a	3, 0	3		3	

$S \rightarrow G_A A \mid G_B B \mid a \mid b$ $A \rightarrow a \mid S G_A$ $B \rightarrow b \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow b$ слово $abba$

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for j=1..n  \ \ j=4
  for k=j-1..1  \ \ k=3
    for all  $A \rightarrow BC \in R$   \ \ если второй нетерминал
      правой части есть в строке 4 с индексом 3
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
      \ \ тогда добавляем в ячейку 4-ой строки для левого
        нетерминала содержимое ячейки первого нетерминала в
        строке 3

```

Подходящие правила:

$S \rightarrow G_A A$, $A \rightarrow S G_A$,
 однако ячейка G_A в
 третьей строке пуста.
 Добавляем содержимое
 ячейки S в ячейку A в
 четвёртой строке.

	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b	1		1, 0		1
3 — b	2, 1		2, 1		2
4 — a	3, 0	3, 2, 1		3	

$S \rightarrow G_A A \mid G_B B \mid a \mid b$ $A \rightarrow a \mid S G_A$ $B \rightarrow b \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow b$ слово abba

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for j=1..n  \ \ j=4
  for k=j-1..1  \ \ k=2
    for all  $A \rightarrow BC \in R$   \ \ если второй нетерминал
      правой части есть в строке 4 с индексом 2
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
      \ \ тогда добавляем в ячейку 4-ой строки для левого
      нетерминала содержимое ячейки первого нетерминала в
      строке 2
    
```

Подходящее правило:
 $S \rightarrow G_A A$, однако ячейка
 G_A во второй строке
 пуста. На этом шаге
 таблица не меняется.

	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b	1		1, 0		1
3 — b	2, 1		2, 1		2
4 — a	3, 0	3, 2, 1		3	

$S \rightarrow G_A A \mid G_B B \mid a \mid b$ $A \rightarrow a \mid S G_A$ $B \rightarrow b \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow b$ слово $abba$

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for  $j=1..n$   $\backslash \backslash j=4$ 
  for  $k=j-1..1$   $\backslash \backslash k=1$ 
    for all  $A \rightarrow BC \in R$      $\backslash \backslash$  если второй нетерминал
      правой части есть в строке 4 с индексом 1
      if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
 $\backslash \backslash$  тогда добавляем в ячейку 4-ой строки для левого
нетерминала содержимое ячейки первого нетерминала в
строке 1

```

Подходящее правило
 опять $S \rightarrow G_A A$, и на
 сей раз нужна ячейка
 G_A не пуста. То, что
 теперь $0 \in T'_4[S]$, показы-
 вает, что $abba \in L(G)$,
 поскольку $T'_4[S] =$
 $\{i \mid a_{i+1} \dots a_4 \in L_G(S)\}$,
 где $a_1 a_2 a_3 a_4 = abba$.

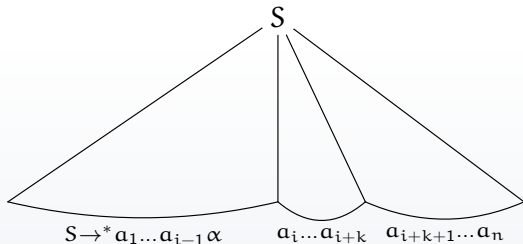
	S	A	B	G_A	G_B
1 — a	0	0		0	
2 — b	1		1, 0		1
3 — b	2, 1		2, 1		2
4 — a	3, 0	3, 2, 1		3	



Нисходящий разбор

Пусть PDA-анализатор хранит пару $\langle \alpha, a_i \dots a_{i+k} \rangle$, где α — сент. форма, $a_i \dots a_{i+k}$ — k следующих символов в строке.

- Если $\alpha = A\alpha'$, тогда по правилу $A \rightarrow \beta$ стековый анализатор переходит в $\langle \beta\alpha', a_i \dots a_{i+k} \rangle$, либо сообщает об ошибке.
- Если $\alpha = a_i\alpha'$, тогда a_i одновременно снимается со стека и читается во входной строке.





LL(k)-грамматики

Определение

Грамматика G называется LL(k) (*left-to-right, leftmost derivation*) \Leftrightarrow в ситуации, когда существуют выводы

$$S \rightarrow^* w_1 A \alpha \rightarrow w_1 \xi \alpha \rightarrow^* w_1 c w_2,$$

$$S \rightarrow^* w_1 A \beta \rightarrow w_1 \eta \beta \rightarrow^* w_1 c w_3, \text{ причём } c \in \Sigma^k,$$

$$w_1, w_2, w_3 \in \Sigma^*, \text{ или } c \in \Sigma^{<k}, w_2 = w_3 = \varepsilon, \text{ всегда } \eta = \beta.$$



LL(k)-грамматики

Определение

Грамматика G называется LL(k) (*left-to-right, leftmost derivation*) \Leftrightarrow в ситуации, когда существуют выводы

$$S \rightarrow^* w_1 A \alpha \rightarrow w_1 \xi \alpha \rightarrow^* w_1 c w_2,$$

$$S \rightarrow^* w_1 A \beta \rightarrow w_1 \eta \beta \rightarrow^* w_1 c w_3, \text{ причём } c \in \Sigma^k,$$

$$w_1, w_2, w_3 \in \Sigma^*, \text{ или } c \in \Sigma^{<k}, w_2 = w_3 = \varepsilon, \text{ всегда } \eta = \beta.$$

Неформально: неоднозначность в выборе правила грамматики при разборе сверху вниз устраняется заглядыванием вперёд на k букв.



Критерий LL(k)-грамматики

Определим

- $\text{FIRST}_k(\eta) = \{a_1 \dots a_j \mid (j < k \ \& \ \eta \rightarrow a_1 \dots a_j) \vee (j = k \ \& \ \eta \rightarrow a_1 \dots a_k \alpha)\} \cup \{\varepsilon \mid \eta \rightarrow^* \varepsilon\}$
- $\text{FOLLOW}_k(\eta) = \{a_1 \dots a_j \mid (j < k \ \& \ S \rightarrow^* \beta \eta a_1 \dots a_j) \vee (j = k \ \& \ S \rightarrow^* \beta \eta a_1 \dots a_k \alpha)\} \cup \{\$ \mid S \rightarrow^* \beta \eta\}$

Тогда G — LL(k)-грамматика $\Leftrightarrow \forall A \rightarrow \alpha, A \rightarrow \beta$
 $(\text{FIRST}_k(\alpha \text{ FOLLOW}_k(A)) \cap \text{FIRST}_k(\beta \text{ FOLLOW}_k(A))) = \emptyset$.



Вычисление FIRST_k

- $\text{FIRST}_k(A) = \{a_1 \dots a_k \mid A \rightarrow a_1 \dots a_k \alpha\} \cup \{a_1 \dots a_j \mid j < k \text{ \& } A \rightarrow a_1 \dots a_j\};$
- До исчерпания: $\forall A \rightarrow B_1 \dots B_n, \text{FIRST}_k(A) = \text{FIRST}_k(A) \cup \text{first}_k(\text{FIRST}_k(B_1) \dots \text{FIRST}_k(B_n))$, где first_k — это k первых символов строки.



Вычисление FIRST_k

- $\text{FIRST}_k(A) = \{a_1 \dots a_k \mid A \rightarrow a_1 \dots a_k \alpha\} \cup \{a_1 \dots a_j \mid j < k \text{ \& } A \rightarrow a_1 \dots a_j\};$
- До исчерпания: $\forall A \rightarrow B_1 \dots B_n, \text{FIRST}_k(A) = \text{FIRST}_k(A) \cup \text{first}_k(\text{FIRST}_k(B_1) \dots \text{FIRST}_k(B_n))$, где first_k — это k первых символов строки.

Алгоритм задаёт фундированный порядок на множестве конфигураций $\text{FIRST}_k(A_i)$, поэтому рано или поздно множества FIRST_k перестанут изменяться.



Вычисление FOLLOW_k

Добавляем правило из нового стартового символа $S_0 \rightarrow S\$$.

- $\text{FOLLOW}_k(A) = \emptyset$ для всех $A \neq S$.
- До исчерпания: $\forall B \rightarrow \beta$ и разбиений $\beta = \eta_1 A \eta_2$,
 $\text{FOLLOW}_k(A) =$
 $\text{FOLLOW}_k(A) \cup \text{first}_k(\text{FIRST}_k(\eta_2) \text{FOLLOW}_k(B))$, где
 first_k — это k первых символов строки.

Алгоритм также задаёт wfo на множестве конфигураций $\text{FOLLOW}_k(A_i)$.



Таблица LL(k)-разбора

Таблица $T_k(A, x)$ по нетерминалу A и lookahead-символам x вычисляет правило для парсинга.

```
for all  $A \rightarrow \alpha$ 
  for all  $x \in \text{first}_k(\text{FIRST}_k(\alpha) \text{ FOLLOW}_k(A))$ 
    if  $T_k(A, x)$  не задано, тогда  $T_k(A, x) = A \rightarrow \alpha$ 
    else объявление о конфликте
```



LL(k)-грамматики без ε -правил

Пример

Стандартный алгоритм удаления ε -правил приводит к разрушению LL-свойств:

$$\begin{aligned} S &\rightarrow ABC \\ A &\rightarrow aA \mid d \\ B &\rightarrow b \mid \varepsilon \\ C &\rightarrow c \mid \varepsilon \end{aligned}$$



LL(k)-грамматики без ε -правил

Утверждение (Куроки–Суонио)

Всякая LL(k)-грамматика может быть преобразована в LL(k+1)-грамматику без ε -правил.



LL(k)-грамматики без ε -правил

Утверждение (Куроки–Суонио)

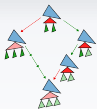
Всякая LL(k)-грамматика может быть преобразована в LL(k+1)-грамматику без ε -правил.

Идея доказательства

Сначала избавимся от всех правил, начинающихся с nullable-нетерминала, путём введения *potnull*-двойников.

Затем присоединим все nullable-нетерминальные отрезки, стоящие в правых частях, к предшествующим им *potnull*-нетерминалам, и объявим полученные строки новыми нетерминалами.

В новых правилах nullable-кусочек приписывается к самому последнему нетерминалу в правой части.

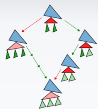


Пример устранения ε без разрушения LL-свойства

$$\begin{array}{ll} S \rightarrow ABC | Bk & A \rightarrow aA | d \\ B \rightarrow b | \varepsilon & C \rightarrow c | \varepsilon \end{array}$$

Сначала избавимся от обнуляемого нетерминала в начале правой части правила стандартным приёмом:

$$\begin{array}{lll} S \rightarrow ABC | B'k | k & A \rightarrow aA | d \\ B \rightarrow b | \varepsilon & C \rightarrow c | \varepsilon & B' \rightarrow b \end{array}$$

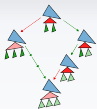


Пример устранения ε без разрушения LL-свойства

$$\begin{array}{lll} S \rightarrow ABC \mid B'k \mid k & A \rightarrow aA \mid d \\ B \rightarrow b \mid \varepsilon & C \rightarrow c \mid \varepsilon & B' \rightarrow b \end{array}$$

Теперь присоединим правый обнуляемый контекст к A и протащим его по правилу переписывания для A :

$$\begin{array}{ll} S \rightarrow [ABC] \mid B'k \mid k & [ABC] \rightarrow a[ABC] \mid [dBC] \\ B \rightarrow b \mid \varepsilon & C \rightarrow c \mid \varepsilon \quad B' \rightarrow b \end{array}$$

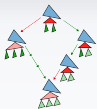


Пример устранения ε без разрушения LL-свойства

$$\begin{array}{ll} S \rightarrow [ABC] \mid B'k \mid k & [ABC] \rightarrow a[ABC] \mid [dBC] \\ B \rightarrow b \mid \varepsilon & C \rightarrow c \mid \varepsilon \quad B' \rightarrow b \end{array}$$

Определяем правила переписывания для нетерминала $[dBC]$, соответствующие коллапсу всего обнуляемого контекста, его префиксов, и непустой развёртке префикса.

$$\begin{array}{ll} S \rightarrow [ABC] \mid B'k \mid k & [ABC] \rightarrow a[ABC] \mid [dBC] \\ [dBC] \rightarrow d[bC] \mid dc \mid d & B \rightarrow b \mid \varepsilon \\ C \rightarrow c \mid \varepsilon & B' \rightarrow b \end{array}$$



Пример устранения ε без разрушения LL-свойства

$$\begin{array}{ll}
 S \rightarrow [ABC] \mid B'k \mid k & [ABC] \rightarrow a[ABC] \mid [dBC] \\
 [dBC] \rightarrow d[bC] \mid dc \mid d & B \rightarrow b \mid \varepsilon \\
 C \rightarrow c \mid \varepsilon & B' \rightarrow b
 \end{array}$$

Аналогично обрабатываем нетерминал $[bC]$ и удаляем все правила для обнуляемых нетерминалов из грамматики.

$$\begin{array}{ll}
 S \rightarrow [ABC] \mid B'k \mid k & [ABC] \rightarrow a[ABC] \mid [dBC] \\
 [dBC] \rightarrow d[bC] \mid dc \mid d & [bC] \rightarrow bc \mid b \quad B' \rightarrow b
 \end{array}$$



Лемма Розенкранца–Стирнса

Утверждение

Если G — $LL(k)$ -грамматика, тогда вышеописанный алгоритм устранения ε -правил всегда завершается, поскольку ни на одном шаге не появляется новых нетерминалов, дважды присоединяющих в правый контекст один и тот же обнуляемый нетерминал грамматики G .

Покажем, что в общем случае алгоритм может и не завершаться. Рассмотрим следующую грамматику, не являющуюся однозначной (и следовательно, $LL(k)$ ни для какого значения k).

$$S \rightarrow SS \mid a \mid \varepsilon$$



Лемма Розенкранца–Стирнса

Утверждение

Если G — $LL(k)$ -грамматика, тогда вышеописанный алгоритм устранения ε -правил всегда завершается.

Покажем, что в общем случае алгоритм может и не завершаться. Рассмотрим следующую грамматику, не являющуюся однозначной (и следовательно, $LL(k)$ ни для какого значения k).

$$S \rightarrow SS \mid \alpha \mid \varepsilon$$

По алгоритму, устраним сначала обнуляемый правый нетерминал.

$$S \rightarrow S'S \mid S' \mid \alpha \mid \varepsilon \quad S' \rightarrow S'S \mid S' \mid \alpha$$

Теперь присоединим обнуляемые контексты (это оставшиеся вхождения S) и посмотрим, какие правила получатся для нового нетерминала $[S'S]$.

$$S \rightarrow [S'S] \mid S' \mid \alpha \mid \varepsilon \quad S' \rightarrow [S'S] \mid S' \mid \alpha \quad [S'S] \rightarrow S'SS \mid S'S \mid \alpha S$$



Лемма Розенкранца–Стирнса

Покажем, что в общем случае алгоритм может и не завершаться. Рассмотрим следующую грамматику, не являющуюся однозначной (и следовательно, $LL(k)$ ни для какого значения k).

$$S \rightarrow SS \mid \alpha \mid \varepsilon$$

По алгоритму, устраним сначала обнуляемый правый нетерминал.

$$S \rightarrow S'S \mid S' \mid \alpha \mid \varepsilon \quad S' \rightarrow S'S \mid S' \mid \alpha$$

Теперь присоединим обнуляемые контексты (это оставшиеся вхождения S) и посмотрим, какие правила получатся для нового нетерминала $[S'S]$.

$$S \rightarrow [S'S] \mid S' \mid \alpha \mid \varepsilon \quad S' \rightarrow [S'S] \mid S' \mid \alpha \quad [S'S] \rightarrow S'SS \mid S'S \mid \alpha S$$

Видно, что нужно порождать новый нетерминал, потому что обнуляемый контекст у правила $[S'S] \rightarrow S'SS$ — это уже два вхождения S . После его порождения опять получим правило вида $[S'SS] \rightarrow S'SSS$, и вообще, для каждого нового $[S'S^n]$ — правило $[S'S^n] \rightarrow S'S^{n+1}$. Значит, ни на какой итерации процесс не сходится.



GNF для $LL(k)$ -грамматики

Утверждения

- Ни одна леворекурсивная грамматика — не $LL(k)$ ни для какого k .
- Всякая $LL(k)$ -грамматика может быть преобразована в $LL(k+1)$ -грамматику в GNF.



LL(k)-языки

Определение

CFL L — это LL(k)-язык, если для него существует LL(k)-грамматика.



LL(k)-языки

Определение

CFL L — это $LL(k)$ -язык, если для него существует $LL(k)$ -грамматика.

Существуют $LL(k)$, но не $LL(k-1)$ -языки.

Рассмотрим язык $\{a^n(b^k d \mid b \mid c)^n\}$. Это $LL(k)$ -язык:

$$\begin{aligned} S &\rightarrow aCA \\ C &\rightarrow aCA \mid \varepsilon \\ A &\rightarrow bB \mid c \\ B &\rightarrow b^{k-1}d \mid \varepsilon \end{aligned}$$

Неформально: не $LL(k-1)$ потому, что иначе бы имелась $LL(k)$ -грамматика в GNF для L . В стеке разбора для префикса a^n оказалось бы больше, чем $2k-1$ символов. Подставляя варианты суффиксов к a^{n+k-1} , получаем противоречие (метод подмены).



Метод подмены

Общая схема метода

- Рассматриваем сразу $LL(k)$ -грамматику в GNF.
- Показываем, что в её стеке в состоянии α должно находиться не меньше, чем $f(k)$ разных символов при предъявлении тех же самых lookahead k символов.
- После данных k символов предъявляем длинный суффикс, отрезок которого должен контекстно-свободно выводиться из некоторого нетерминала в стеке.
- Предъявляем другой суффикс к тому же префиксу и lookahead, в котором не может быть такого отрезка, и подменяем вывод нетерминала.



Техника метода подмены

- Ищем такие два слова xfy , xfz , что $|f| = k$ и при чтении префикса x стек мог неограниченно разрастись (т.е. есть синхронная накачка хотя бы одной из пар x и y и x и z). Предполагаем высоту стека равной хотя бы $k + t$ (где t выбирается в зависимости от задачи).
- k символов в стеке при выталкивании точно распознают фрагмент f (lookahead). Рассматриваем, что может распознаться остальными t символами. Комбинируем распознанные фрагменты и показываем, что их комбинация не входит в язык.
- Либо если в стеке осталось t символов, а длина, например, y меньше t , тогда с этим стеком точно нельзя распознать y .



Пример подмены

Может ли язык $\{a^n b^n\} \cup \{a^n c^n\}$ задаваться LL(k)-грамматикой для некоторого k ?

Пусть такое k нашлось (без ограничения общности — в GNF). Рассмотрим слово $a^{n+k} b^{n+k}$ и подберём такое n , что после чтения a^n LL-анализатор имеет в стеке не меньше $k + 2$ символов. Пусть последний символ — это Y . Он распознаёт некоторое подслово суффикса b^{n+k} , а именно b^s . Теперь подменим строку на $a^{n+k} c^{n+k}$. В этом случае Y должен распознать некоторое c^t . Но значит, $a^{n+k} b^{n+k-s} c^t \in L$, что невозможно. □



Dangling Else

Рассмотрим GNF-грамматику для языка $\{a^n b^m \mid n \geq m\}$.



Dangling Else

Рассмотрим GNF-грамматику для языка $\{a^n b^m \mid n \geq m\}$. Положим $w = a^{n+k} b^{n+k}$, где n таково, что в стеке на момент чтения a^n не меньше, чем $k + 2$ нетерминала. Тогда при подмене w на a^{n+k+1} из стека достанется только k нетерминалов, и слово a^{n+k+1} распознаться не сможет.



Dangling Else

Рассмотрим GNF-грамматику для языка $\{a^n b^m \mid n \geq m\}$. Положим $w = a^{n+k} b^{n+k}$, где n таково, что в стеке на момент чтения a^n не меньше, чем $k + 2$ нетерминала. Тогда при подмене w на a^{n+k+1} из стека достанется только k нетерминалов, и слово a^{n+k+1} распознаться не сможет.

Следствие

Dangling else не парсится с помощью LL-разбора.



Свойства $LL(k)$ -языков

Утверждение

- $LL(k)$ -языки не замкнуты относительно объединения.
- $LL(k)$ -языки не замкнуты относительно пересечения с регулярным языком (пример: $\{a^n b a^n b\} \cup \{a^n c a^n c\}$ — не $LL(k)$ -язык).
- Если L — $LL(k)$ - нерегулярный язык, то его дополнение — не $LL(k)$ ни для какого k .