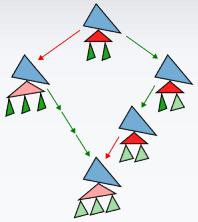
Нисходящий разбор

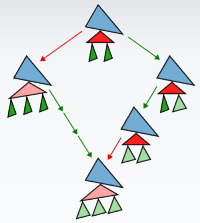
Теория формальных языков

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.

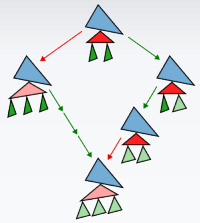


Неоднозначные КС-языки

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.



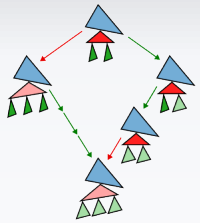
Неоднозначные КС-языки

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

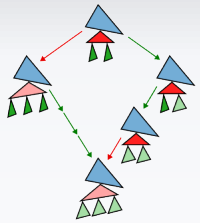
КС-грамматика G неоднозначная, если существует слово $w \in L(G)$ такое, что в G у него больше одного дерева разбора. КС-язык L существенно неоднозначен, если всякая его грамматика неоднозначна.

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 $T'_j[A] = \{i \mid a_{i+1} \dots a_j \in L_G(A)\}$.

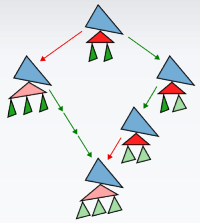


Алгоритм Касами–Тории

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 CT

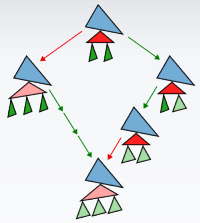
Читаем очередной символ, вычисляем множество $T'_j[A]$ для всех $A \in N$, постепенно достраивая его как список, упорядоченный по возрастанию. Если $A \to BC$, тогда если $k \in T'_j[C]$, то для всех $x \in T'_k[B]$ выполнено $x \in T'_j[A]$.



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 $T'_j[A] = \{i \mid a_{i+1} \dots a_j \in L_G(A)\}$.

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n
  for all $A \in N$ if $A \to a_j \in R$ then $T'_j[A] = \{j-1\}$
  for k=j-1..1
    for all $A \to BC \in R$
      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 $T'_j[A] = \{i \mid a_{i+1} \dots a_j \in L_G(A)\}$.

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n
  for all $A \in N$ if $A \to a_j \in R$ then $T'_j[A] = \{j-1\}$
  for k=j-1..1
    for all $A \to BC \in R$
      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 $O(n^2)$.

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

$\forall \mathbf{j}, \mathbf{A}(\mathbf{T}'_{\mathbf{j}}[\mathbf{A}] = \varnothing)$

```
for j=1..n
  for k=j-1..1
    for all A → BC ∈ R
      if k ∈ T'_j[C] then for all i ∈ T'_k[B]  T'_j[A] = T'_j[A] ∪ {i}
```

Инициализация таблицы:

	S	A	B	G_A	G_B
$1 - \mathbf{a}$					
$2 - \mathbf{b}$					
$3 - \mathbf{b}$					
$4 - \mathbf{a}$					

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

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n \\ $\mathbf{j = 1}$
  for all $A \in N$ **if** $A \to a_j \in R$ **then** $T'_j[A] = \{j - 1\}$
  for k=j-1..1
    for all $A \to BC \in R$
      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 правилам, цикла по нетерминальным нет, т.к. $k = 0$:

	S	A	B	G_A	G_B
$1 - \mathbf{a}$	0	0		0	
$2 - \mathbf{b}$					
$3 - \mathbf{b}$					
$4 - \mathbf{a}$					

$S \to G_A A \mid G_B B \mid a \mid \mathbf{b}$ $A \to a \mid SG_A$ $B \to \mathbf{b} \mid SG_B$
$G_A \to a$ $G_B \to \mathbf{b}$ слово $abba$

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n \\ **j = 2**
  for all $A \in N$ **if** $A \to a_j \in R$ **then** $T'_j[A] = \{j-1\}$
  for k=j-1..1
    for all $A \to BC \in R$
      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 правилам:

	S	A	B	G_A	G_B
$1 - \mathbf{a}$	0	0		0	
$2 - \mathbf{b}$	1		1		1
$3 - \mathbf{b}$					
$4 - \mathbf{a}$					

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

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n \\ $\mathbf{j = 2}$
  for k=j-1..1 \\ $\mathbf{k = 1}$
    for all $A \to BC \in R$ \\ 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 $S \to G_B B$) ячейка в первой строке пуста, так что добавляем только содержимое ячейки для S в ячейку B.

	S	A	B	G_A	G_B
$1 - \mathbf{a}$	0	0		0	
$2 - \mathbf{b}$	1		1, 0		1
$3 - \mathbf{b}$					
$4 - \mathbf{a}$					

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

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n \\ $\mathbf{j = 3}$
  for all $A \in N$ **if** $A \to a_j \in R$ **then** $T'_j[A] = \{j - 1\}$
  for k=j-1..1
    for all $A \to BC \in R$
      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 правилам:

	S	A	B	G_A	G_B
$1 - \mathbf{a}$	0	0		0	
$2 - \mathbf{b}$	1		1, 0		1
$3 - \mathbf{b}$	2		2		2
$4 - \mathbf{a}$					

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

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n \\ **j = 3**
  for k=j-1..1 \\ **k = 2**
    for all $A \to BC \in R$ \\ 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: $S \to G_B B$, $B \to S G_B$, 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				

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

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n \\ **j = 3**
  for k=j-1..1 \\ **k = 1**
    for all $A \to BC \in R$ \\ 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 бывает вторым в правой части. Остаётся правило $S \to G_B B$, 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				

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

$\forall j, A(T'_j[A] = \varnothing)$
```
for j=1..n \\ j = 4
  for all A ∈ N if A → a_j ∈ R then T'_j[A] = {j − 1}
  for k=j-1..1
    for all A → BC ∈ R
      if k ∈ T'_j[C] then for all i ∈ T'_k[B]  T'_j[A] = T'_j[A] ∪ {i}
```

$j = 4$, проход по терминальным правилам:

	S	A	B	G_A	G_B
$1 - \mathbf{a}$	0	0		0	
$2 - \mathbf{b}$	1		1, 0		1
$3 - \mathbf{b}$	2, 1		2, 1		2
$4 - \mathbf{a}$	3, 0	3		3	

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

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n \\ $\mathbf{j = 4}$
  for k=j-1..1 \\ $\mathbf{k = 3}$
    for all $A \to BC \in R$ \\ 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: $S \to G_A A$, $A \to S G_A$, 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	

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

$\forall j, A(T'_j[A] = \varnothing)$
for j=1..n \\ **j = 4**
  for k=j-1..1 \\ **k = 2**
    for all $A \to BC \in R$ \\ 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: $S \to G_A A$, однако ячейка G_A во второй строке пуста. На этом шаге таблица не меняется.

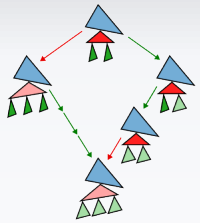
	S	A	B	G_A	G_B
1 — **a**	0	0		0	
2 — **b**	1		1, 0		1
3 — **b**	2, 1		2, 1		2
4 — **a**	3, 0	3, 2, 1		3	

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

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

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

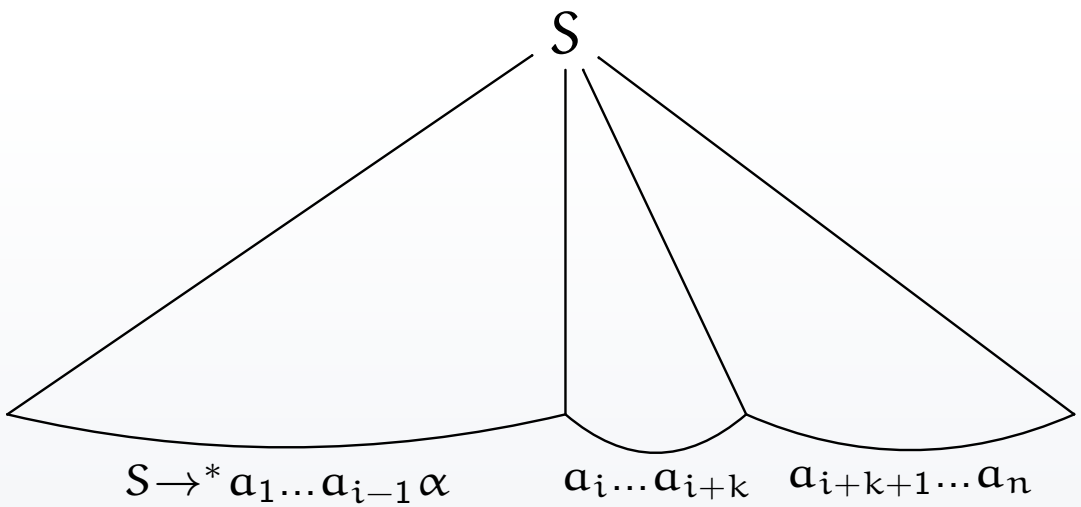
	S	A	B	G_A	G_B
1 — **a**	0	0		0	
2 — **b**	1		1, 0		1
3 — **b**	2, 1		2, 1		2
4 — **a**	3, 0	3, 2, 1		3	

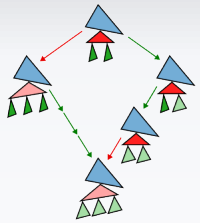


Нисходящий разбор

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 $A \to \beta$ 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.

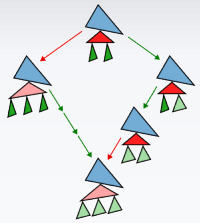




LL(k)-грамматики

Определение

Грамматика G называется LL(k) *(left-to-right, leftmost derivation)* $\Leftrightarrow$ в ситуации, когда существуют выводы
$S \to^* w_1 A \alpha \to w_1 \xi \alpha \to^* w_1 c w_2$,
$S \to^* w_1 A \beta \to w_1 \eta \beta \to^* w_1 c w_3$, причём $c \in \Sigma^k$,
$w_1, w_2, w_3 \in \Sigma^*$, или $c \in \Sigma^{<k}$, $w_2 = w_3 = \varepsilon$, всегда $\eta = \beta$.

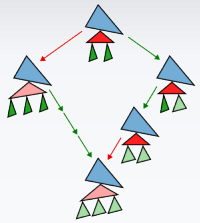


LL(k)-грамматики

Определение

Грамматика G называется LL(k) *(left-to-right, leftmost derivation)* $\Leftrightarrow$ в ситуации, когда существуют выводы
$S \to^* w_1 A \alpha \to w_1 \xi \alpha \to^* w_1 c w_2$,
$S \to^* w_1 A \beta \to w_1 \eta \beta \to^* w_1 c w_3$, причём $c \in \Sigma^k$,
$w_1, w_2, w_3 \in \Sigma^*$, или $c \in \Sigma^{<k}$, $w_2 = w_3 = \varepsilon$, всегда $\eta = \beta$.

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

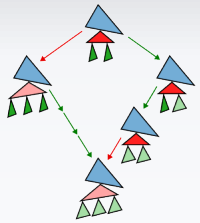


Критерий LL(k)-грамматики

Определим

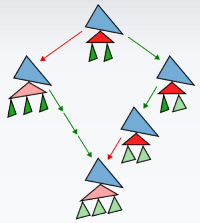
- $\mathrm{FIRST}_k(\eta) = \{a_1 \dots a_j \mid (j < k \,\&\, \eta \to a_1 \dots a_j) \vee (j = k \,\&\, \eta \to a_1 \dots a_k \alpha)\} \cup \{\varepsilon \mid \eta \to^* \varepsilon\}$
- $\mathrm{FOLLOW}_k(\eta) = \{a_1 \dots a_j \mid (j < k \,\&\, S \to^* \beta\eta a_1 \dots a_j) \vee (j = k \,\&\,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

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



Вычисление FIRST_k

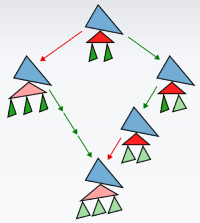
- $\mathsf{FIRST}_k(A) = \{a_1 \dots a_k \mid A \to a_1 \dots a_k \alpha\}$ $\cup \{a_1 \dots a_j \mid j < k \;\&\; A \to a_1 \dots a_j\}$;
- До исчерпания: $\forall A \to B_1 \dots B_n$, $\mathsf{FIRST}_k(A) = \mathsf{FIRST}_k(A) \cup first_k(\mathsf{FIRST}_k(B_1) \dots \mathsf{FIRST}_k(B_n))$, где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



Вычисление FIRST_k

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

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



Вычисление FOLLOW_k

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

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

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

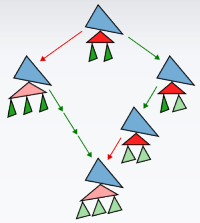
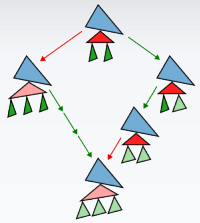


Таблица LL(k)-разбора

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

```
for all A → α
  for all x ∈ first_k(FIRST_k(α) FOLLOW_k(A))
    if T_k(A, x) не задано, тогда T_k(A, x) = A → α
       else объявление о конфликте
```

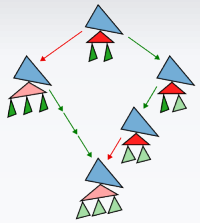


LL(k)-грамматики без ε-правил

Пример

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

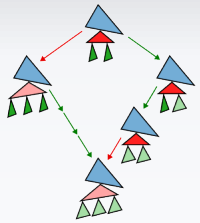
$$\begin{aligned} S &\to ABC \\ A &\to aA \mid d \\ B &\to b \mid \varepsilon \\ C &\to c \mid \varepsilon \end{aligned}$$



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)

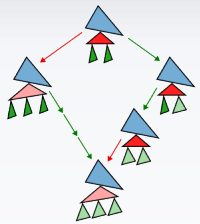
Всякая LL(k)-грамматика может быть преобразована в LL(k+1)-грамматику без ε-правил.

Идея доказательства

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

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

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

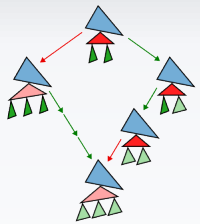


Пример устранения ε без разрушения LL-свойства

$$S \to ABC \mid Bk \quad A \to aA \mid d$$
$$B \to b \mid \varepsilon \qquad C \to c \mid \varepsilon$$

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

$$S \to ABC \mid B'k \mid k \quad A \to aA \mid d$$
$$B \to b \mid \varepsilon \qquad C \to c \mid \varepsilon \qquad B' \to b$$

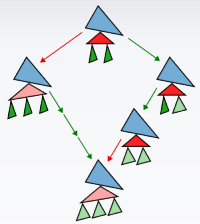


Пример устранения ε без разрушения LL-свойства

$$S \to ABC \,|\, B'k \,|\, k \quad A \to aA \,|\, d$$
$$B \to b \,|\, \varepsilon \quad C \to c \,|\, \varepsilon \quad B' \to b$$

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

$$S \to [ABC] \,|\, B'k \,|\, k \quad [ABC] \to a[ABC] \,|\, [dBC]$$
$$B \to b \,|\, \varepsilon \quad C \to c \,|\, \varepsilon \quad B' \to b$$

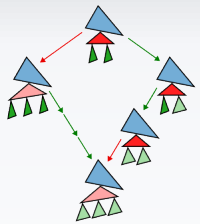


Пример устранения ε без разрушения LL-свойства

$$S \to [ABC] \mid B'k \mid k \qquad [ABC] \to a[ABC] \mid [dBC]$$
$$B \to b \mid \varepsilon \qquad C \to c \mid \varepsilon \qquad B' \to b$$

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

$$S \to [ABC] \mid B'k \mid k \qquad [ABC] \to a[ABC] \mid [dBC]$$
$$[dBC] \to d[bC] \mid dc \mid d \qquad B \to b \mid \varepsilon$$
$$C \to c \mid \varepsilon \qquad B' \to b$$

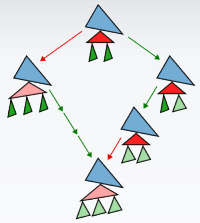


Пример устранения ε без разрушения LL-свойства

$$S \to [ABC] \mid B'k \mid k \qquad [ABC] \to a[ABC] \mid [dBC]$$
$$[dBC] \to d[bC] \mid dc \mid d \qquad B \to b \mid \varepsilon$$
$$C \to c \mid \varepsilon \qquad B' \to b$$

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

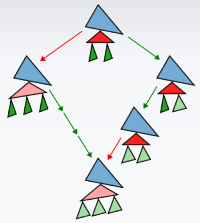
$$S \to [ABC] \mid B'k \mid k \qquad [ABC] \to a[ABC] \mid [dBC]$$
$$[dBC] \to d[bC] \mid dc \mid d \qquad [bC] \to bc \mid b \quad B' \to b$$



GNF для LL(k)-грамматики

Утверждения

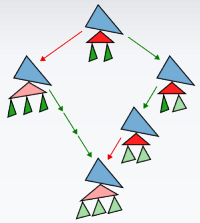
- Ни одна леворекурсивная грамматика — не LL(k) ни для какого k.
- Всякая LL(k)-грамматика может быть преобразована в LL(k+1)-грамматику в GNF.



LL(k)-языки

Определение

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



LL(k)-языки

Определение

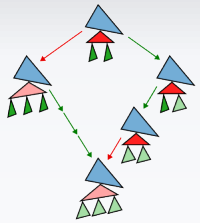
CFL L — это LL(k)-язык, если для него существует LL(k)-грамматика.

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:

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

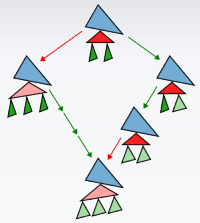
Неформально: не LL(k-1) потому, что иначе бы имелась LL(k)-грамматика в GNF для L. В стеке разбора для префикса a^n оказалось бы больше, чем $2k-1$ символов. Подставляя варианты суффиксов к a^{n+k-1}, получаем противоречие (метод подмены).



Метод подмены

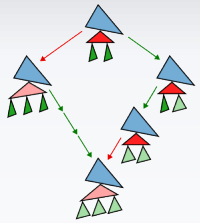
Общая схема метода

- 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

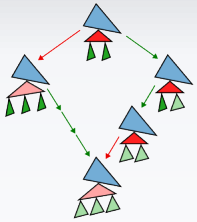
- Ищем такие два слова xfy, xfz, что $|\mathrm{f}| = \mathrm{k}$ и при чтении префикса x стек мог неограниченно разрастись (т.е. есть синхронная накачка хотя бы одной из пар x и y и x и z). Предполагаем высоту стека равной хотя бы $\mathrm{k} + \mathrm{t}$ (где t выбирается в зависимости от задачи).
- 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

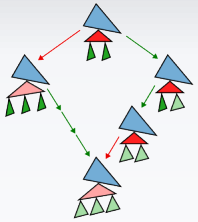
Может ли язык $\{a^n b^n\} \cup \{a^n c^n\}$ задаваться LL(k)-грамматикой для некоторого k?

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 распознает 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. □



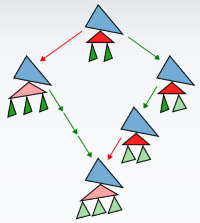
Dangling Else

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



Dangling Else

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

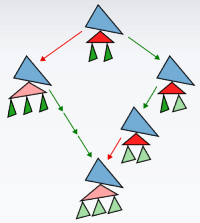


Dangling Else

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