

Восходящий разбор. Неоднозначность и детерминированность



Теория формальных языков
2022 г.



(Не)лирическое отступление





(Не)лирическое отступление

- Дополнительные задания не требуют больших объёмов кода, доступны всем (в том числе группам) и стоят минимум 2 балла. Следим за горящими сроками!
- Запись на доделывание закрывается после дедлайна.
- Планируем действия заранее:
 - 3 лабораторная по объёму большая — используем чужой код (с конкурса, от прошлых цивилизаций), но не забываем про структуры (и Рефал-стайл).
 - 4 лабораторная будет сложной (и прошлые цивилизации помогут мало). Зато будет 2 допзадания!
 - 5 лабораторная будет легче, но выполняться в мини-группах. Допзадание будет одно.
 - На последней неделе(!) будет биг-фарма: возможность добрать баллы по всем темам решением задач.



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.



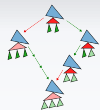
Неоднозначные КС-языки

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

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

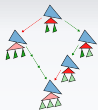


Алгоритм Касами–Тории

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  $A \rightarrow \alpha_j \in R$  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 правилам, цикла по нетерминальным 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  $j=1..n \setminus j = 2$ 
  for all  $A \in N$  if  $A \rightarrow \alpha_j \in R$  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  $j=1..n \setminus j=3$ 
  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					

$S \rightarrow G_A A \mid G_B B \mid a \mid b$ $A \rightarrow a \mid S G_A$ $B \rightarrow b \mid S G_B$
 $G_A \rightarrow a$ $G_B \rightarrow b$ 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 бывает 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					

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

```

 $\forall j, A(T'_j[A] = \emptyset)$ 
for  $j=1..n \setminus j=4$ 
  for all  $A \in N$  if  $A \rightarrow \alpha_j \in R$  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	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 содержи-
 мое 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	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	3, 2, 1		3	

$$S \rightarrow G_A A \mid G_B B \mid a \mid b \quad A \rightarrow a \mid S G_A \quad B \rightarrow b \mid S G_B$$

$$G_A \rightarrow a \quad G_B \rightarrow b \quad \text{слово } abba$$

```

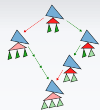
 $\forall j, A(T'_j[A] = \emptyset)$ 
for j=1..n \ \ j = 4
  for k=j-1..1 \ \ k = 1
    for all  $A \rightarrow BC \in R$  \ \ если второй нетерминал
      правой части есть в строке 4 с индексом 1
        if  $k \in T'_j[C]$  then for all  $i \in T'_k[B]$   $T'_j[A] = T'_j[A] \cup \{i\}$ 
  \ \ тогда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

```

Подходящее правило опять

$S \rightarrow G_A A$, 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	



Детерминированные КС-языки

Язык L обладает префикс-свойством (prefix-free), если $\forall w(w \in L \Rightarrow \forall v(v \neq \varepsilon \Rightarrow wv \notin L))$.



Детерминированные КС-языки

Язык L обладает префикс-свойством (prefix-free), если $\forall w(w \in L \Rightarrow \forall v(v \neq \varepsilon \Rightarrow wv \notin L))$.

Детерминированные языки с префикс-свойством — языки, распознаваемые DPDA с допуском по пустому стеку.

Рассмотрим язык a^+ . Предположим, он распознаётся DPDA с допуском по пустому стеку. Тогда на элементе a стек уже обязательно пуст. А значит, работа DPDA не может быть продолжена, и элемент aa не может быть им распознан.



Детерминированные КС-языки

Язык L обладает префикс-свойством (prefix-free), если $\forall w(w \in L \Rightarrow \forall v(v \neq \varepsilon \Rightarrow wv \notin L))$.

Детерминированные языки с префикс-свойством — языки, распознаваемые DPDA с допуском по пустому стеку.

Рассмотрим язык L , $w_1, w_1w_2 \in L$, $w_2 \neq \varepsilon$. Предположим, он распознаётся DPDA с допуском по пустому стеку. Тогда на элементе w_1 стек уже обязательно пуст. А значит, работа DPDA не может быть продолжена, и элемент w_1w_2 не может быть им распознан.



Эндмаркеры

Рассмотрим язык $a^+\$$ (алфавит терминалов $\Sigma = \{a, \$\}$). В этом языке ни одно слово не является префиксом другого.



Эндмаркеры

Рассмотрим язык $\{w\$ \mid w \in L\}$ (алфавит терминалов $\Sigma = \Sigma_L \cup \{\$, \$ \notin \Sigma_L\}$). Независимо от L , в этом языке ни одно слово не является префиксом другого.

- Хорошие новости: любой детерминированный КС-язык легко преобразовать в язык, распознаваемый DPDA с допуском по пустому стеку.



Эндмаркеры

Рассмотрим язык $\{w\$ \mid w \in L\}$ (алфавит терминалов $\Sigma = \Sigma_L \cup \{\$, \$ \notin \Sigma_L\}$). Независимо от L , в этом языке ни одно слово не является префиксом другого.

- Хорошие новости: любой детерминированный КС-язык легко преобразовать в язык, распознаваемый DPDA с допуском по пустому стеку.
- Плохие новости: существенно неоднозначные контекстно-свободные языки с префикс-свойством. Стандартный пример: $\{a^n b^n c^m d\} \cup \{a^m b^n c^n d\}$.



Языки нередуцируемых префиксов

Определим понятие свёртки — перехода справа налево в правиле переписывания $A \rightarrow \alpha$. Что можно сказать о всех возможных префиксах сентенциальных форм, порождаемых грамматикой G , к которым нельзя применить ни одну свёртку?



Языки нередуцируемых префиксов

Определим понятие свёртки — перехода справа налево в правиле переписывания $A \rightarrow \alpha$. Что можно сказать о всех возможных префиксах сентенциальных форм, порождаемых грамматикой G , к которым нельзя применить ни одну свёртку?

Такие с.ф. образуют регулярный язык. Идея обоснования: в распознающем их PDA из стека ничего не читается, т.е. PDA учитывает только символы сент. формы и свои состояния.



Автомат нередуцируемых префиксов

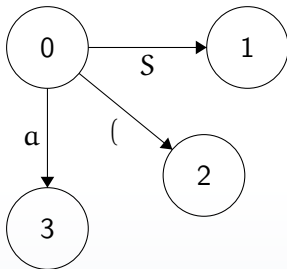
Описание конструкции

- Отмеченная позиция в правиле: \bullet . В правиле с правой частью $\xi_1 \dots \xi_n$ есть $n + 1$ таких позиций.
- Правило $A \rightarrow \alpha \bullet B \beta$ и правило $B \rightarrow \bullet \gamma$ — одно и то же множество переходов по символу, не приводящих к редукции \Rightarrow в одном состоянии.
- При чтении элемента правой части сдвигаем \bullet вправо на позицию.



Автомат нередуцируемых префиксов

$S' \rightarrow S \quad S \rightarrow a \quad S \rightarrow (S)$

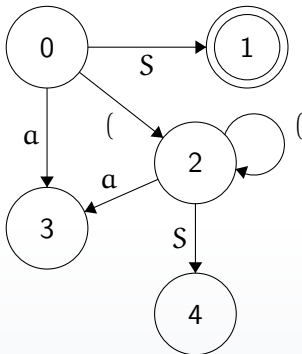


0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$
1	$S' \rightarrow S \bullet$
2	$S \rightarrow (\bullet S)$
3	$S \rightarrow a \bullet$



Автомат нередуцируемых префиксов

$S' \rightarrow S \quad S \rightarrow a \quad S \rightarrow (S)$

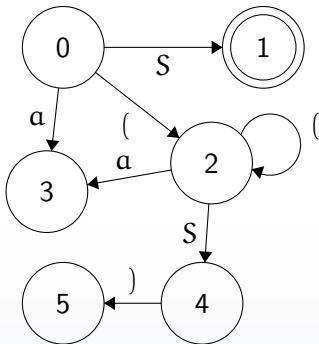


0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$
1	$S' \rightarrow S \bullet$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$
3	$S \rightarrow a \bullet$
4	$S \rightarrow (S \bullet)$

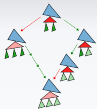


Автомат нередуцируемых префиксов

$S' \rightarrow S \quad S \rightarrow a \quad S \rightarrow (S)$



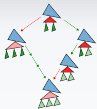
0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$
1	$S' \rightarrow S \bullet$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$
3	$S \rightarrow a \bullet$
4	$S \rightarrow (S \bullet)$
5	$S \rightarrow (S) \bullet$



Типы состояний автомата

- ❶ Финальное (свёртка в S').
- ❷ Не финальное, но свёртка.
- ❸ Сдвиг по символу сентенциальной формы.

Что хранить в стеке PDA, построенного по такому автомату?

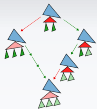


Типы состояний автомата

- 1 Финальное (свёртка в S').
- 2 Не финальное, но свёртка.
- 3 Сдвиг по символу сентенциальной формы.

Что хранить в стеке PDA, построенного по такому автомату?

- Хранить сами сентенциальные формы плохо — проблема с извлечением нескольких подряд символов.



Типы состояний автомата

- 1 Финальное (свёртка в S').
- 2 Не финальное, но свёртка.
- 3 Сдвиг по символу сентенциальной формы.

Что хранить в стеке PDA, построенного по такому автомату?

- Хранить сами сентенциальные формы плохо — проблема с извлечением нескольких подряд символов.
- Логично хранить последовательности последних символов с.ф., которые могут привести к разным свёрткам, закодированными одним символом стека.



Типы состояний автомата

- ❶ Финальное (свёртка в S').
- ❷ Не финальное, но свёртка.
- ❸ Сдвиг по символу сентенциальной формы.

Что хранить в стеке PDA, построенного по такому автомату?

- Хранить сами сентенциальные формы плохо — проблема с извлечением нескольких подряд символов.
- Логично хранить последовательности последних символов с.ф., которые могут привести к разным свёрткам, закодированными одним символом стека.
- А это — в точности состояния автомата.



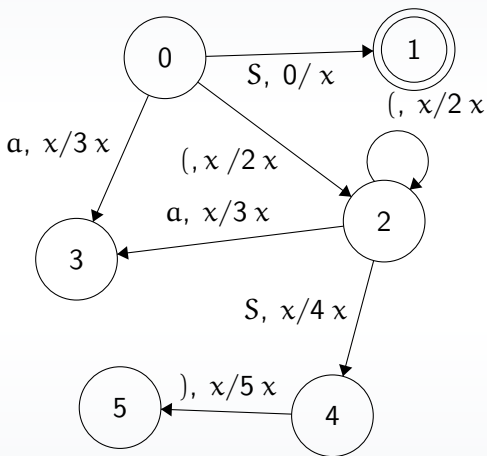
PDA по LR(0)-автомату

Общая конструкция

- При каждом сдвиге кладём в стек номер состояния, в которое приходим в конечном автомате.
- При каждой свёртке извлекаем из стека n символов, где n — длина правой части β правила $A \rightarrow \beta$, после чего переходим в состояние с номером $n + 1$ -ого символа в стеке, подразумевая на ленте символ A .
- Совершаем переход по символу A из полученного состояния (этот шаг мы на графе объединили с предыдущим).



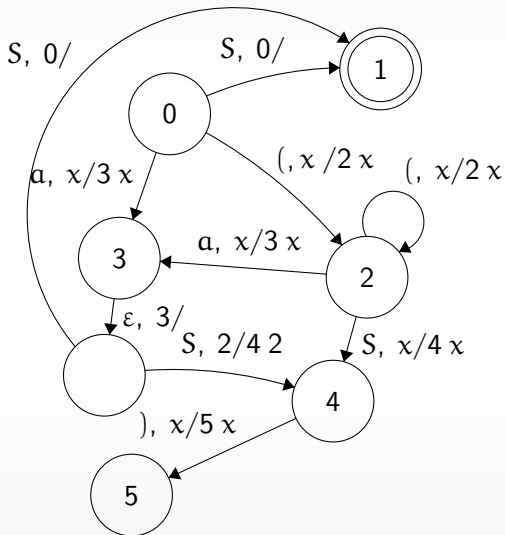
Пример построения PDA



0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
1	$S' \rightarrow S \bullet$	$S \rightarrow S'$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
3	$S \rightarrow a \bullet$	$a \rightarrow S$
4	$S \rightarrow (S \bullet)$	
5	$S \rightarrow (S) \bullet$	$(S) \rightarrow S$



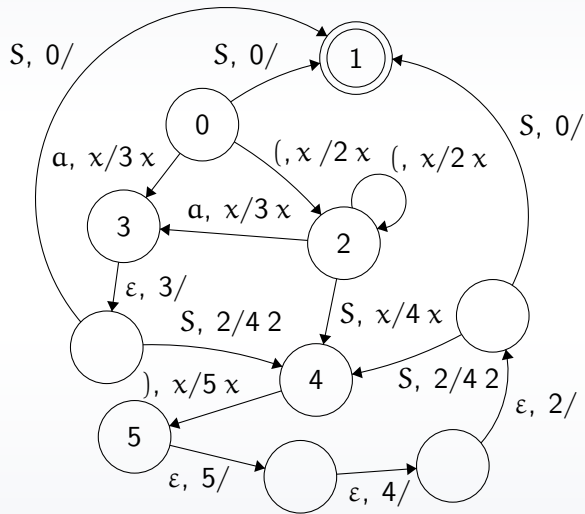
Пример построения PDA



0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
1	$S' \rightarrow S \bullet$	$S \rightarrow S'$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
3	$S \rightarrow a \bullet$	$a \rightarrow S$
4	$S \rightarrow (S \bullet)$	
5	$S \rightarrow (S) \bullet$	$(S) \rightarrow S$



Пример построения PDA

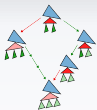


0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
1	$S' \rightarrow S \bullet$	$S \rightarrow S'$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
3	$S \rightarrow a \bullet$	$a \rightarrow S$
4	$S \rightarrow (S \bullet)$	
5	$S \rightarrow (S) \bullet$	$(S) \rightarrow S$

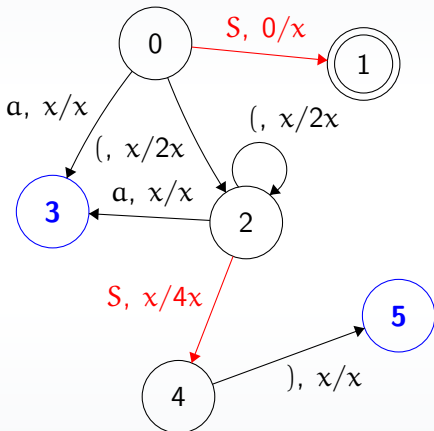


Промежуточный PDA-распознаватель

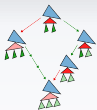
- Стековые символы, ведущие в состояния свёртки, не являющиеся финальными (у нас это 3 и 5), бесполезны, потому что сразу же безальтернативно извлекаются из стека.
- Распознаватель ещё не может быть использован как парсер, потому что он «читает» нетерминалы с ленты. Этого можно избежать, если принять, что нетерминал обязан быть считанным сразу после свёртки, и объединить свёртку (порождение нетерминала) и его считывание в один ε -переход.
- После добавления таких ε -переходов исходные переходы по нетерминалам можно удалять.



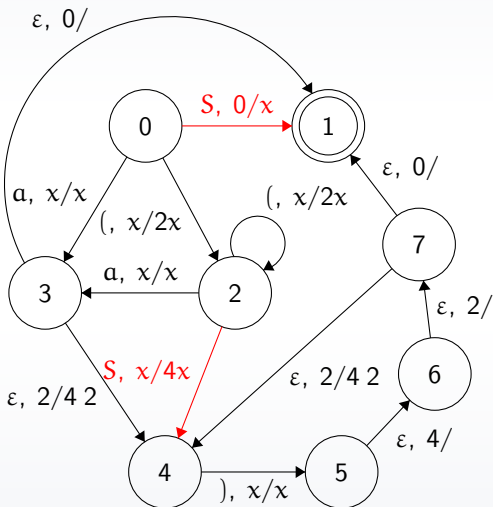
Избавление от переходов по нетерминалам



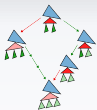
0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
1	$S' \rightarrow S \bullet$	$S \rightarrow S'$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
3	$S \rightarrow a \bullet$	$a \rightarrow S$
4	$S \rightarrow (S \bullet)$	
5	$S \rightarrow (S) \bullet$	$(S) \rightarrow S$



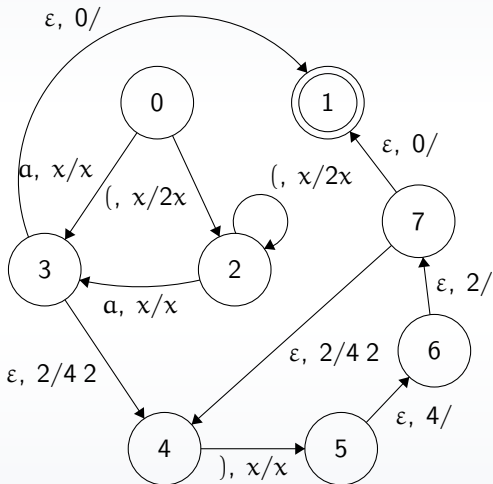
Избавление от переходов по нетерминалам



0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
1	$S' \rightarrow S \bullet$	$S \rightarrow S'$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
3	$S \rightarrow a \bullet$	$a \rightarrow S$
4	$S \rightarrow (S \bullet)$	
5	$S \rightarrow (S) \bullet$	$(S) \rightarrow S$



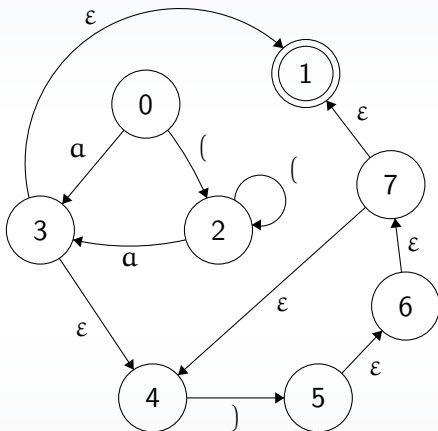
Избавление от переходов по нетерминалам



0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
1	$S' \rightarrow S \bullet$	$S \rightarrow S'$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$	
3	$S \rightarrow a \bullet$	$a \rightarrow S$
4	$S \rightarrow (S \bullet)$	
5	$S \rightarrow (S) \bullet$	$(S) \rightarrow S$



Бонус — регулярная аппроксимация



Аппроксимацией исходного языка $(a^n)^n$, построенной по LR(0)-автомату (Pereira–Wright), является язык $(a^*)^*$.



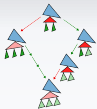
PDA или DPDA?

- Если есть ε -переходы, то нет никаких других.
- Если есть ε -переход, то он единственный из данного состояния.



PDA или DPDA?

- Если есть ϵ -переходы, то нет никаких других. Если делается свёртка, то нельзя сделать сдвиг.
- Если есть ϵ -переход, то он единственный из данного состояния. Если делается свёртка одного типа, то нельзя сделать свёртку другого типа.
- Допуск — по пустому стеку \Rightarrow DPDA для языков с префикс-свойством.
- DPDA с допуском по пустому стеку распознают те же языки, что и LR(0)-разбор.
- В конструкции LR(0)-автомата часто навязывается эндмаркер \Rightarrow изначальная грамматика может описывать не LR(0)-язык!



Отказ от эндмаркера и SLR

- Используем **ту же конструкцию автомата**.
- Разрешим при возможности сделать свёртку вида $\beta \rightarrow A$ заглянуть в множество $FOLLOW(A)$, чтобы понять, какую свёртку делать (и делать ли).



Отказ от эндмаркера и SLR

- Используем **ту же конструкцию автомата**.
- Разрешим при возможности сделать свёртку вида $\beta \rightarrow A$ заглянуть в множество $\text{FOLLOW}(A)$, чтобы понять, какую свёртку делать (и делать ли).

$$\begin{array}{lll} S' \rightarrow E & E \rightarrow E + T & E \rightarrow T \\ E \rightarrow V = E & T \rightarrow (E) & T \rightarrow \text{id} \\ & V \rightarrow \text{id} & \end{array}$$

Здесь есть конфликт свёрток для S' (по $V \rightarrow \text{id}\bullet$ и $T \rightarrow \text{id}\bullet$), но $\text{FOLLOW}_1(V) \cap \text{FOLLOW}_1(T) = \emptyset \Rightarrow$ эта грамматика — SLR(1).



Коллапс линейных парсеров

Теорема

Для всякого языка из класса DCFL существует распознающая его $\text{SLR}(1)$ -грамматика.



Теоретический коллапс линейных парсеров

Теорема

Для всякого языка из класса DCFL существует распознающая его $SLR(1)$ -грамматика.

Следует из теоремы:

Для всякого языка из класса DCFL существует распознающая его $LR(k)$ -грамматика.



LR(k)-распознаватели

Грамматика G — LR(k), тогда и только тогда, когда для всех пар сентенциальных форм xu, xu' , порождаемых правосторонним разбором, где $u, u' \in \Sigma^+$, таких что xu допускает правую свёртку в префиксе u по правилу ξ_1 , а xu' — свёртку где угодно по правилу ξ_2 , и первые k символов u и u' совпадают, $\xi_1 = \xi_2$.



LR(k)-распознаватели

Грамматика G — LR(k), тогда и только тогда, когда для всех пар сентенциальных форм $xу$, $xу'$, порождаемых правосторонним разбором, где $у, у' \in \Sigma^+$, таких что $xу$ допускает правую свёртку в префиксе $у$ по правилу ξ_1 , а $xу'$ — свёртку где угодно по правилу ξ_2 , и первые k символов $у$ и $у'$ совпадают, $\xi_1 = \xi_2$.

$$\begin{array}{lll} S' \rightarrow S & S \rightarrow L = R; & S \rightarrow R; \\ L \rightarrow \text{id} & L \rightarrow *R & R \rightarrow L \end{array}$$

Поскольку $= \in \text{FOLLOW}_1(R)$, возникает конфликт вида сдвиг–свёртка при попытке анализа с.ф. L . Но lookahead у L , порождённой посредством $S \rightarrow L = R$, и посредством $S \rightarrow R; \rightarrow L;$, будет разный.



LR(k)-распознаватели

Грамматика G — LR(k), тогда и только тогда, когда для всех пар сентенциальных форм xu, xu' , порождаемых правосторонним разбором, где $u, u' \in \Sigma^+$, таких что xu допускает правую свёртку в префиксе u по правилу ξ_1 , а xu' — свёртку где угодно по правилу ξ_2 , и первые k символов u и u' совпадают, $\xi_1 = \xi_2$.

Любая LR(k)-грамматика по определению гарантирует однозначный разбор при определённой длине lookahead-строки, поэтому ни одна грамматика с неоднозначным разбором не является LR(k) ни для какого значения k .



$LR(k) \rightarrow LR(1)$, Mickunas–Lancaster–Shneider

$$\begin{array}{lll} S' \rightarrow S & S \rightarrow Abb & S \rightarrow Bbc \\ A \rightarrow aA & A \rightarrow a & B \rightarrow aB \\ & B \rightarrow a & \end{array}$$

Не $LR(1)$, из-за свёрток $A \rightarrow a$, $B \rightarrow a$. Используем трансформацию присоединения правого контекста:

$$\begin{array}{lll} S' \rightarrow S & S \rightarrow [Ab]b & S \rightarrow [Bb]c \\ [Ab] \rightarrow a[Ab] & [Ab] \rightarrow ab & [Bb] \rightarrow a[Bb] \\ & [Bb] \rightarrow ab & \end{array}$$



LR(k) \rightarrow LR(1), Mickunas–Lancaster–Shneider

$$\begin{aligned} S' &\rightarrow S & S &\rightarrow bSS & S &\rightarrow a \\ & & S &\rightarrow aac \end{aligned}$$

Не LR(1), конфликт свёртки на префиксе ba с контекстом a .

Используем трансформацию уточнения правого контекста:

$$\begin{aligned} S &\rightarrow bSa[a/S] & S &\rightarrow bSb[b/S] & S &\rightarrow a & S &\rightarrow aac \\ [a/S] &\rightarrow \varepsilon & [a/S] &\rightarrow ac & [b/S] &\rightarrow Sa[a/S] & [b/S] &\rightarrow Sb[b/S] \end{aligned}$$

Теперь присоединим правые контексты:

$$\begin{array}{llll} S \rightarrow b[Sa][a/S] & | b[Sb][b/S] & | a & | aac \\ [Sa] \rightarrow b[Sa][[a/S]a] & | b[Sb][[b/S]a] & | aa & | aaca \\ [Sb] \rightarrow b[Sa][[a/S]b] & | b[Sb][[b/S]b] & | ab & | aacb \\ [a/S] \rightarrow \varepsilon & | ac & & \\ [[a/S]a] \rightarrow a & | aca & & \\ [[a/S]b] \rightarrow b & | acb & & \\ [[b/S]a] \rightarrow [Sa][[a/S]a] & | [Sb][[b/S]a] & & \\ [[b/S]b] \rightarrow [Sa][[a/S]b] & | [Sb][[b/S]b] & & \end{array}$$



Применение MLS-подгонки

Исследовать на детерминированность язык
 $L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$

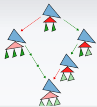
Видно, что если язык L распознаётся DPDA (т.е. является LR(1)-языком), то он также является LR(0)-языком, поскольку удовлетворяет префикс-свойству. Действительно, любое слово этого языка содержит единственную букву c , причём она расположена точно в середине слова.

Построим пробную КС-грамматику для языка L :

$$S \rightarrow aSb \mid aCa \mid bCb \mid c$$

$$C \rightarrow aCa \mid bCb \mid c$$

Проверим, является ли она LR(0)-грамматикой. Для этого построим LR(0)-автомат и проанализируем его на конфликты.



Применение MLS-подгонки

Исследовать на детерминированность язык

$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

Пробная грамматика для L:

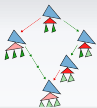
$$\begin{array}{ll} S & \rightarrow aSb \mid aCa \mid bCb \mid c \\ C & \rightarrow aCa \mid bCb \mid c \end{array}$$

Начинаем строить LR(0)-автомат. Для этого вводим новое стартовое состояние S' (состояние окончательной свёртки) и начинаем разбор правила $S' \rightarrow \bullet S$.

Поскольку отмеченная позиция в правиле находится перед нетерминалом S , добавляем в состояние все ситуации вида $S \rightarrow \bullet \alpha$.

Переходы по нетерминалу S и терминалу c ведут к бесконфликтным свёрткам, поэтому малоинтересны. Разберёмся с переходом по a .

$S' \rightarrow \bullet S$
 $S \rightarrow \bullet aSb$
 $S \rightarrow \bullet aCa$
 $S \rightarrow \bullet bCb$
 $S \rightarrow \bullet c$



Применение MLS-подгонки

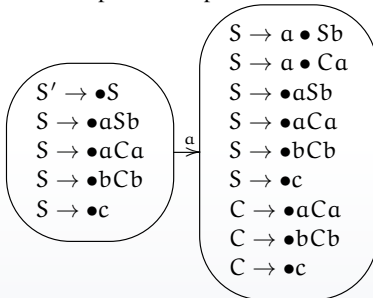
Исследовать на детерминированность язык

$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

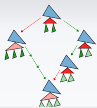
Пробная грамматика для L :

$$\begin{array}{lcl} S & \rightarrow & aSb \mid aCa \mid bCb \mid c \\ C & \rightarrow & aCa \mid bCb \mid c \end{array}$$

Переходы по нетерминалу S и терминалу c ведут к бесконфликтным свёрткам, поэтому малоинтересны. Разберёмся с переходом по a .



Похоже, что есть потенциальный конфликт (даже два) по свёрткам в S и C . Построим конфликтное состояние явно.



Применение MLS-подгонки

Исследовать на детерминированность язык

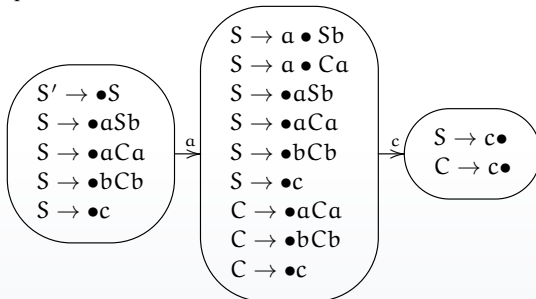
$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

Пробная грамматика для L :

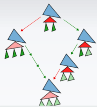
$$\begin{array}{lcl} S & \rightarrow & aSb \mid aCa \mid bCb \mid c \\ C & \rightarrow & aCa \mid bCb \mid c \end{array}$$

Похоже, что есть потенциальный конфликт (даже два) по свёрткам в S и C .

Построим конфликтное состояние явно.



Присоединим к конфликтующим S и C -нетерминалам их правые контексты.



Применение MLS-подгонки

Исследовать на детерминированность язык

$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

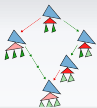
Пробная грамматика для L :

$$\begin{array}{lcl} S & \rightarrow & aSb \mid aCa \mid bCb \mid c \\ C & \rightarrow & aCa \mid bCb \mid c \end{array}$$

Грамматика для L после присоединения правых контекстов к нетерминалам S и C методом MLS (новые нетерминалы выделены красным):

$$\begin{array}{lcl} S & \rightarrow & a[Sb] \mid a[Ca] \mid b[Cb] \mid c \\ [Sb] & \rightarrow & a[Sb]b \mid a[Ca]b \mid b[Cb]b \mid cb \\ [Ca] & \rightarrow & a[Ca]a \mid b[Cb]a \mid ca \\ [Cb] & \rightarrow & a[Ca]b \mid b[Cb]b \mid cb \end{array}$$

Можно построить LR(0)-автомат для этой грамматики и убедиться, что он не содержит конфликтов. Значит язык L — детерминированный (более того, LR(0)).

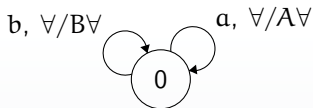


Другой подход к анализу КС-языков

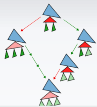
Исследовать на детерминированность язык

$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

Можно сразу попробовать построить DPDA для L . Заметим, что до прочтения буквы с стек обязательно заполняется (иначе потеряется информация либо о структуре палиндрома, либо о количестве букв a в начале слова), причём, поскольку неизвестно, когда именно префикс a^n переходит в палиндром, придётся запоминать, какие конкретные буквы были прочитаны: считаем, что символ стека A соответствует a , символ B — терминалу b .



Для экономии места символ \forall использован в роли параметра, пробегающего значения A , B и Z_0 : на детерминированность это не влияет, поскольку переходы с его участием делаются по разным терминалам.

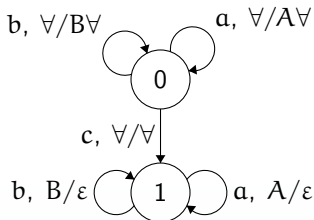


Другой подход к анализу КС-языков

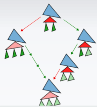
Исследовать на детерминированность язык

$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

После прочтения буквы c стек только опустошается: структура оставшейся части слова определяется уже прочитанной его частью.



Единственная тонкость — это переход от чтения w^R к чтению b^n . Он происходит, если на вершине стека лежит A , а читается буква b , и это не приводит к неопределённости, поскольку при чтении буквы b из палиндромной части мы обязаны всегда иметь на вершине стека символ B .

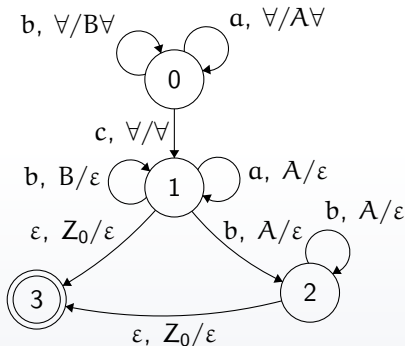


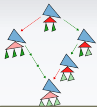
Другой подход к анализу КС-языков

Исследовать на детерминированность язык

$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

Добавляем состояние чтения суффикса b^n (в нём на вершине стека должны быть всегда лишь символы A) и финальное состояние. Легко убедиться, что итоговый стековый автомат — DPDA.



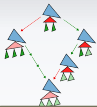


Лемма о накачке для DCFL

Теорема (S. Yu)

Пусть L — DCFL. Тогда существует такая длина накачки p , что для всех пар слов $w, w' \in L$, таких что $w = xy$ & $w' = xz$, $|x| > p$ и первые буквы y, z совпадают, выполнено одно из двух:

- 1 существует накачка только префикса x (в привычном смысле);
- 2 существует разбиение $x = x_1x_2x_3$, $y = y_1y_2y_3$, $z = z_1z_2z_3$ такое, что $|x_2x_3| \leq p$, $|x_2| > 0$, и $\forall i (x_1x_2^ix_3y_1y_2^iy_3 \in L \text{ \& } x_1x_2^ix_3z_1z_2^iz_3 \in L)$.



Лемма о накачке для DCFL

Теорема (S. Yu)

Пусть L — DCFL. Тогда существует такая длина накачки p , что для всех пар слов $w, w' \in L$, таких что $w = xy$ & $w' = xz$, $|x| > p$ и первые буквы y, z совпадают, выполнено одно из двух:

- 1 существует накачка только префикса x (в привычном смысле);
- 2 существует разбиение $x = x_1x_2x_3$, $y = y_1y_2y_3$, $z = z_1z_2z_3$ такое, что $|x_2x_3| \leq p$, $|x_2| > 0$, и $\forall i (x_1x_2^ix_3y_1y_2^iy_3 \in L \text{ \& } x_1x_2^ix_3z_1z_2^iz_3 \in L)$.

Рассмотрим язык $\{a^n b^n\} \cup \{a^n b^{2n}\}$, положим $x = a^n b^{n-1}$, $y = b$, $z = b^{2n-1}$, где $n - 1 > p$. Тогда в случае 2 придётся накачивать в x только b , а в случае 1 нет подходящей накачки.



Замыкания DCFL

- Замкнуты относительно дополнения (смена конечных состояний в DPDA).
- Замкнуты относительно пересечения с регулярным языком.
- Не замкнуты относительно объединения (см. $\{a^n b^n\} \cup \{a^n b^{2n}\}$).
- Не замкнуты относительно пересечения.



Замыкания DCFL

- Замкнуты относительно дополнения (смена конечных состояний в DPDA).
- Замкнуты относительно пересечения с регулярным языком.
- Не замкнуты относительно объединения (см. $\{a^n b^n\} \cup \{a^n b^{2n}\}$).
- Не замкнуты относительно пересечения.
- Не замкнуты относительно гомоморфизмов. См. $\{ca^n b^n\} \cup \{a^n b^{2n}\}$.
- Не замкнуты относительно конкатенации. См. $L_1 = \{ca^n b^n\} \cup \{a^n b^{2n}\}$, $L_2 = c^*$.



Метод подмены vs накачка для DCFL

- Так же, как и в лемме о накачке для DCFL, нужно подобрать два слова xu , xz с длинными одинаковыми префиксами и различными суффиксами u , z , принадлежащие языку L .
- В лемме о накачке суффиксы u и z должны иметь существенно разное происхождение с точки зрения их распознавания PDA (разное поведение стека на префиксе x в слове xu и в слове xz), а в методе подмены часто достаточно, если стек на x только накапливается, а на u и z читается по-разному.
- В обоих случаях x лучше выбирать так, чтобы от поведения стека на нём максимально сильно зависел успех распознавания суффиксов u и z .



Метод подмены vs накачка для DCFL

Исследовать $LL(k)$ -свойства уже известного нам DCFL

$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

- Подберём два слова с одинаковым поведением стека до буквы c и разными суффиксами. Проще всего это сделать, если положить, что до c встречаются только буквы a . Тогда $xz = a^{n+k} c a^{n+k}$, $yz = a^{n+k} c b^{n+k}$, где n так велико, что после прочтения префикса $x = a^n$ в стеке точно есть минимум $k + 3$ символа, где k — предполагаемый lookahead.



Метод подмены vs накачка для DCFL

Исследовать $LL(k)$ -свойства уже известного нам DCFL

$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

- $xz = a^{n+k} c a^{n+k}$, $yz = a^{n+k} c b^{n+k}$, где n так велико, что после прочтения префикса $x = a^n$ в стеке точно есть минимум $k + 3$ символа, где k — предполагаемый lookahead.
- Пусть последний символ стека после чтения a^n — T_z . В слове $a^{n+k} c a^{n+k}$ при чтении символа T_z анализатору будет видно $k_1 \leq k$ букв a (если $k_1 < k$, то за ними будет конец слова), и начиная с этого состояния анализатор распознает суффикс a^i , $i \geq k_1$. В слове $a^{n+k} c b^{n+k}$ при чтении символа T_z анализатор увидит $k_2 \leq k$ букв b и распознает суффикс b^j , $j \geq k_2$.
- Если заменить в слове $a^{n+k} c b^{n+k}$ суффикс b^j на a^i , то анализатор прочтёт T_z с lookahead'ом, равным a^{k_1} . Ситуация ничем не будет отличаться от той, где он видел a^{k_1} букв в суффиксе слова $a^{n+k} c a^{n+k}$, и анализатор определит, что слово $a^{n+k} c b^{n+k-j} a^i \in L$, что неверно. Значит, L — не $LL(k)$.



Степень недетерминизма языка

Если PDA \mathcal{A} допускает декомпозицию на DPDA, между которыми есть максимум k недетерминированных переходов, но не допускает такую декомпозицию при $i < k$ переходов, скажем, что \mathcal{A} задаёт КС-язык с k -недетерминированностью.



Степень недетерминизма языка

Если PDA \mathcal{A} допускает декомпозицию на DPDA, между которыми есть максимум k недетерминированных переходов, но не допускает такую декомпозицию при $i < k$ переходов, скажем, что \mathcal{A} задаёт КС-язык с k -недетерминированностью.

- 1 Степень недетерминированности языка $\{a^n b^n\} \cup \{a^n b^{2n}\}$?



Степень недетерминизма языка

Если PDA \mathcal{A} допускает декомпозицию на DPDA, между которыми есть максимум k недетерминированных переходов, но не допускает такую декомпозицию при $i < k$ переходов, скажем, что \mathcal{A} задаёт КС-язык с k -недетерминированностью.

- 1 Степень недетерминированности языка $\{a^n b^n\} \cup \{a^n b^{2n}\}$?
Ответ: 1
- 2 Степень недетерминированности языка $\{a^n b^n\} \cup \dots \cup \{a^n b^{k \cdot n}\}$?



Степень недетерминизма языка

Если PDA \mathcal{A} допускает декомпозицию на DPDA, между которыми есть максимум k недетерминированных переходов, но не допускает такую декомпозицию при $i < k$ переходов, скажем, что \mathcal{A} задаёт КС-язык с k -недетерминированностью.

- 1 Степень недетерминированности языка $\{a^n b^n\} \cup \{a^n b^{2n}\}$?
Ответ: 1
- 2 Степень недетерминированности языка $\{a^n b^n\} \cup \dots \cup \{a^n b^{k \cdot n}\}$? Ответ: тоже 1 (см. критерий исправляемости)
- 3 Степень недетерминированности языка $\{ww^R\}$ — также 1.
- 4 Степень недетерминированности языка $\{ww^R vv^R\}$ равна 2.



Исправление недетерминированности

Пусть L — недетерминированный КС-язык и $k > 0$. Язык L — k -исправляемый, если существует алфавит Δ , $\Delta \cap \Sigma = \emptyset$ и DCFL $L(k) \subseteq (\Sigma \cup \Delta)^*$ такой, что для $h(\Delta) = \varepsilon$, $h(L(k)) = L$ и все слова языка $L(k)$ содержат не больше k букв из Δ .

Язык L имеет k -ую степень недетерминизма $\Leftrightarrow L$ k -исправляемый, но не $k - 1$ -исправляемый.



Исправляемость и анализ на DCFL

Техника использования леммы о накачке для DCFL

- анализируем позиции в словах языка L , в которых может произойти смена наполнения стека на его опустошение, а может не произойти. Такие позиции считаем подозрительными на исправляемость.
- подбираем два слова из L , xuz_1 , xuz_2 такие, что исправляемая позиция находится в подслове u , причём в подслове u слова xuz_1 происходит наполнение стека, а в слове xuz_2 стек опустошается либо игнорируется.
- убеждаемся, что отдельно x накачать нельзя, после чего рассматриваем накачки uz_1 и uz_2 . Из-за разного поведения стека на их префиксах, скорее всего, эти накачки будут выводить из языка L .



Пример применения

Проанализировать контекстно-свободный язык

$$L = \{w a^n c^n w^R \mid w \in \{a, b\}^*\}.$$

- В словах языка есть произвольные подслова из $\{a, b\}^*$, что усложняет анализ. К тому же есть блок c^n , который на первый взгляд однозначно указывает на детерминизм, однако нет условия $n \geq 1$, поэтому в некоторых случаях на его существование нельзя положиться. Воспользуемся замкнутостью DCFL относительно пересечений с регулярными языками, избавимся от c^n и сузим область накачек.
- Простейший язык, с которым мы можем пересечь L для этой цели: $a^* b^* a^*$, после чего взять $xy = a^m$, $z_1 = a^{n_1}$, $z_2 = a^{n_2} b^{2 \cdot n_3} a^{m+n_2}$.



Пример применения

Проанализировать контекстно-свободный язык

$$L = \{w a^n c^n w^R \mid w \in \{a, b\}^*\}.$$

- Нужно избавиться от под слова с буквами c и сузить область накачек.
- Простейший язык, с которым мы можем пересечь L для этой цели: $a^* b^* a^*$, после чего взять $xu = a^m$, $z_1 = a^{n_1}$, $z_2 = a^{n_2} b^{2 \cdot n_3} a^{m+n_2}$.
- Хотя поведение стека на этих фрагментах слов соответствует рекомендуемому, анализ ни к чему не приводит: мы без проблем можем накачивать в этих словах одновременно суффикс послова xu и элементы z_1 и z_2 , а всё потому, что слова в языке a^* , являющиеся палиндромами, описываются регулярными выражениями. Искомое пересечение языков неудачное, выберем то, которое чётче обозначит нерегулярную структуру палиндрома.



Пример применения

Проанализировать контекстно-свободный язык

$$L = \{w a^n c^n w^R \mid w \in \{a, b\}^*\}.$$

- Рассмотрим пересечение L с языком $a^*b^*a^*b^*a^*$. В нём уже будут два типа палиндромов, не распознаваемые регулярами (с одним или двумя подсловами, состоящими из букв b).
- Абеляр (т.е. антагонист) выбирает длину накачки p .
- Элоиза (т.е. мы) выбирает слова $a^{p+1}ba^{p+1}$ и $a^{p+1}ba^{p+1}a^{p+1}ba^{p+1}$ и $xy = a^{p+1}ba^p$, $z_1 = a$, $z_2 = a^{p+2}ba^{p+1}$.
- Абеляр не может накачивать только $a^{p+1}ba^p$: при накачке только второго a^p произойдёт рассинхронизация с суффиксом z_1 , а при любой накачке с участием первого a^{p+1} — рассинхронизация с суффиксом z_2 .
- Значит, Абеляру остаётся только накачивать подслово суффикса a^p синхронно с подсловом z_1 (т.е. a) (и некоторым подсловом z_2 , но это уже не важно), что также приводит к выходу из языка палиндромов.



Пример применения

Проанализировать контекстно-свободный язык

$$L = \{w a^n c^n w^R \mid w \in \{a, b\}^*\}.$$

- Рассмотрим пересечение L с языком $a^*b^*a^*b^*a^*$.
- Абеляр (т.е. антагонист) выбирает длину накачки p .
- Элоиза (т.е. мы) выбирает слова $a^{p+1}ba^{p+1}$ и $a^{p+1}ba^{p+1}a^{p+1}ba^{p+1}$ и $xy = a^{p+1}ba^p$, $z_1 = a$, $z_2 = a^{p+2}ba^{p+1}$.
- Абеляр не может накачивать только $a^{p+1}ba^p$: при накачке только второго a^p произойдёт рассинхронизация с суффиксом z_1 , а при любой накачке с участием первого a^{p+1} — рассинхронизация с суффиксом z_2 .
- Значит, Абеяру остаётся только накачивать подслово суффикса a^p синхронно с подсловом z_1 (т.е. a) (и некоторым подсловом z_2 , но это уже не важно), что также приводит к выходу из языка палиндромов.
- Заметим, что если взять слова a^pba^p и $a^pba^pa^pba^p$ и $xy = a^pba^{p-1}$, тогда синхронную накачку придумать можно: накачивать в xy букву b (она ещё в пределах длины накачки), в z_2 её же, а в z_1 ничего.
- Мы показали, что язык пересечения — NCFL, значит, язык L — NCFL.



Иерархия недетерминированных КС-языков

Семейство языков $w_1 w_1^R \$ \dots w_k w_k^R \$$ ($\$ \notin \Sigma$) задаёт бесконечную иерархию недетерминированных языков с k -недетерминизмом.



Иерархия недетерминированных КС-языков

Семейство языков $w_1 w_1^R \$ \dots w_k w_k^R \$$ ($\$ \notin \Sigma$) задаёт бесконечную иерархию недетерминированных языков с k -недетерминизмом.

- Введение вложенных структур с совпадающими маркерами начала и конца приводит к неограниченному недетерминизму.



Иерархия Хомского revisited

Утверждения ниже касаются только языков (не грамматик)!

- $\text{RegL} \subset \text{CFL}$;
- $\text{RegL} \subset \text{DCFL}$;
- $\text{DCFL} \subset \text{CFL}$;
- $\text{RegL} \subset \text{LL}(1)$;
- $\text{LR}(0)$ не сравним с RegL ;
- $\text{LR}(0)$ не сравним с $\text{LL}(k)$;
- $\text{LL}(k) \subset \text{LL}(k+1)$;
- $\text{LL}(k) \subset \text{LR}(1)$;
- $\text{LR}(k) = \text{SLR}(1) = \text{DCFL}$.

