

Восходящий разбор



Теория формальных языков
2021 г.



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.



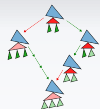
Детерминированные КС-языки

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

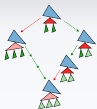
Рассмотрим язык (алфавит терминалов). этом языке ни одно слово не является префиксом другого.

- 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

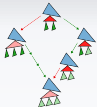
Определим понятие свёртки — перехода справа налево в правиле переписывания $A \rightarrow \alpha$. Что можно сказать о всех возможных префиксах сентенциальных форм, порождаемых грамматикой G , к которым нельзя применить ни одну свёртку?



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

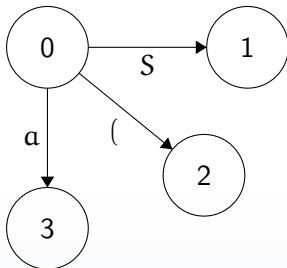
Описание конструкции

- 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

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

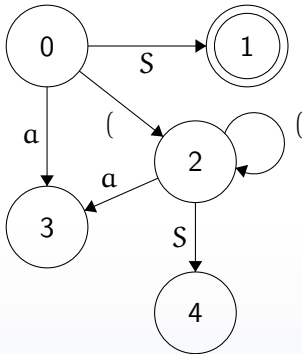


0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$
1	$S' \rightarrow S \bullet$
2	$S \rightarrow (\bullet S)$
3	$S \rightarrow a \bullet$



Автомат нередуцируемых префиксов

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

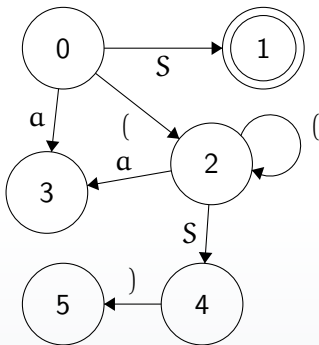


0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet(S)$ $S \rightarrow \bullet a$
1	$S' \rightarrow S \bullet$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet(S)$ $S \rightarrow \bullet a$
3	$S \rightarrow a \bullet$
4	$S \rightarrow (S \bullet)$

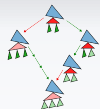


Автомат нередуцируемых префиксов

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



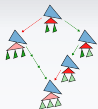
0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$
1	$S' \rightarrow S \bullet$
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet (S)$ $S \rightarrow \bullet a$
3	$S \rightarrow a \bullet$
4	$S \rightarrow (S \bullet)$
5	$S \rightarrow (S) \bullet$



Типы состояний автомата

- 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?



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

- 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.
- А это — в точности состояния автомата.



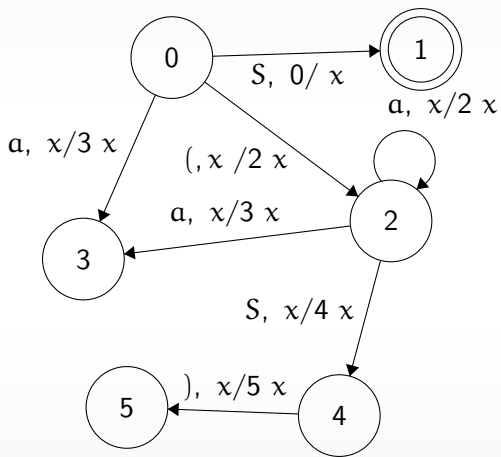
PDA по LR(0)-автомату

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 по символу A .



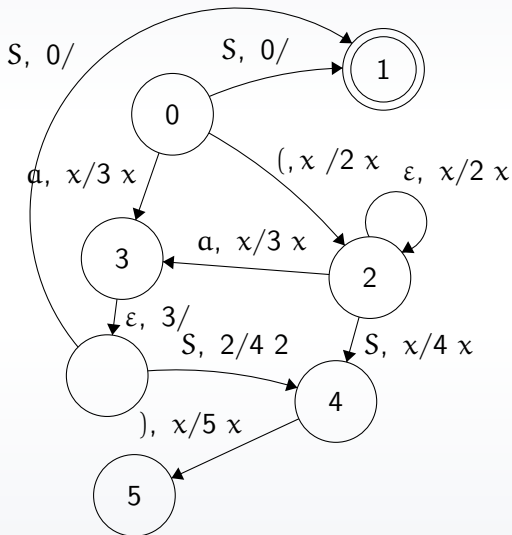
Пример построения PDA



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



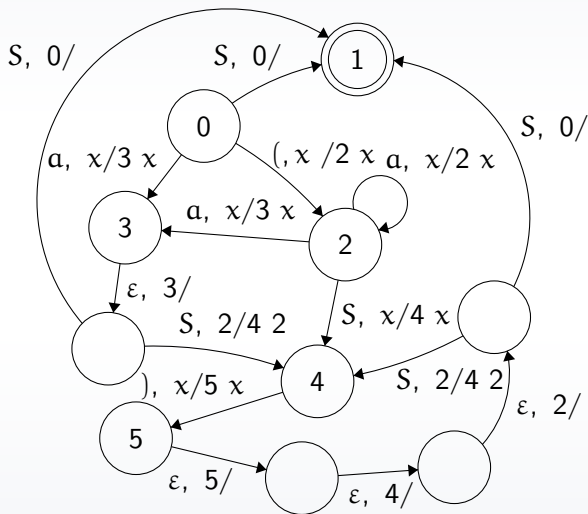
Пример построения PDA



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

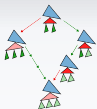


0	$S' \rightarrow \bullet S$ $S \rightarrow \bullet(S)$ $S \rightarrow \bullet a$	
1	$S' \rightarrow S \bullet$ $S \rightarrow S'$	
2	$S \rightarrow (\bullet S)$ $S \rightarrow \bullet(S)$ $S \rightarrow \bullet a$	
3	$S \rightarrow a \bullet$ $a \rightarrow S$	
4	$S \rightarrow (S \bullet)$	
5	$S \rightarrow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



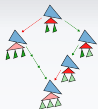
PDA или DPDA?

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



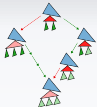
PDA или DPDA?

- 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).

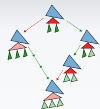


Отказ от эндмаркера и SLR

- 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 $SLR(1)$ -грамматика.



Теоретический коллапс линейных парсеров

Теорема

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

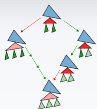
Следует из теоремы:

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 $y, y' \in \Sigma^+$, таких что xu допускает правую свёртку в префиксе y по правилу ξ_1 , а xu' — свёртку где угодно по правилу ξ_2 , и первые k символов y и y' совпадают, $\xi_1 = \xi_2$.



LR(k)-распознаватели

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

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 .

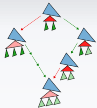


$LR(k) \rightarrow LR(1)$, **Mickunas–Lancaster–Shneider**

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

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:

$S \rightarrow b[Sa][a/S]$	$ 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]$		



Применение MLS-подгонки

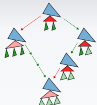
Исследовать на детерминированность язык
 $L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$

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

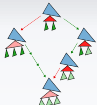
$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

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

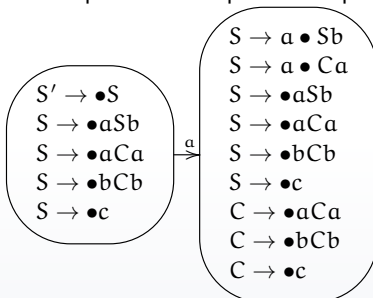
Исследовать на детерминированность язык

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

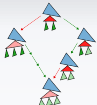
Пробная грамматика для L :

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

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

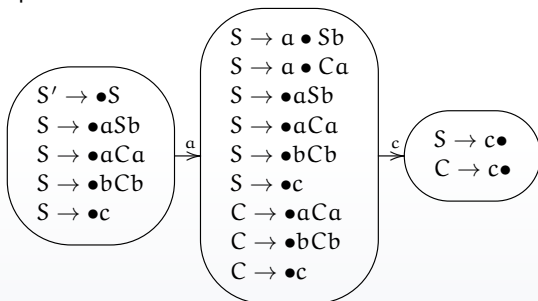
Исследовать на детерминированность язык

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

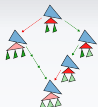
Пробная грамматика для L :

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

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

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

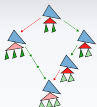
Пробная грамматика для L :

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

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)).

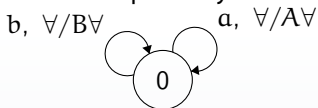


Другой подход к анализу КС-языков

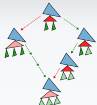
Исследовать на детерминированность язык

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

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.

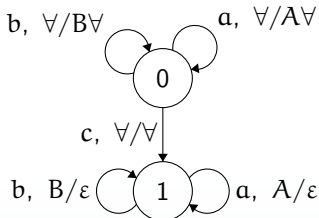


Другой подход к анализу КС-языков

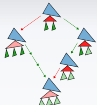
Исследовать на детерминированность язык

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

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 .



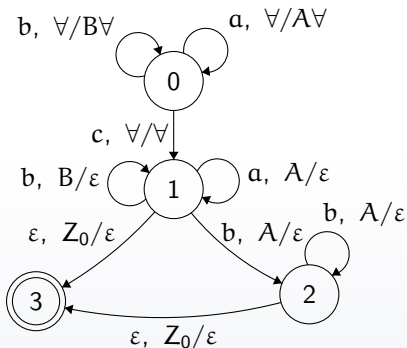
Другой подход к анализу КС-языков

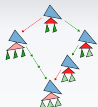
Исследовать на детерминированность язык

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

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.



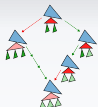


Лемма о накачке для DCFL

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

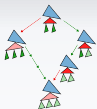
- Так же, как и в лемме о накачке для DCFL, нужно подобрать два слова xu , xz с длинными одинаковыми префиксами и различными суффиксами y , z , принадлежащие языку L .
- В лемме о накачке суффиксы y и z должны иметь существенно разное происхождение с точки зрения их распознавания PDA (разное поведение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



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 $\text{LL}(k)$;
- $\text{LL}(k) \subset \text{LL}(k+1)$;
- $\text{LL}(k) \subset \text{LR}(1)$;
- $\text{LR}(k) = \text{SLR}(1) = \text{DCFL}$.

