Восходящий разбор

Теория формальных языков

2021 г.

Детерминированные КС-языки

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

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 $\forall w (w \in \mathrm{L} \Rightarrow \forall v (v \neq \varepsilon \Rightarrow wv \notin \mathrm{L}))$.

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

Рассмотрим язык L, $w_1, w_1 w_2 \in \mathrm{L}$, $w_2 \neq \varepsilon$. 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 $w_1 w_2$ 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 $A \to \alpha$. 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 $A \to \alpha$. 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.

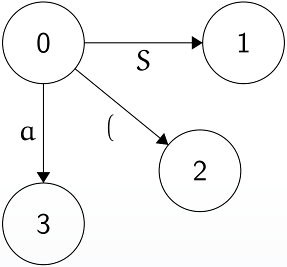
Автомат нередуцируемых префиксов

Описание конструкции

- Отмеченная позиция в правиле: $\bullet$. В правиле с правой частью $\xi_1 \dots \xi_n$ есть $n+1$ таких позиций.
- Правило $A \to \alpha \bullet B\beta$ и правило $B \to \bullet\gamma$ — 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

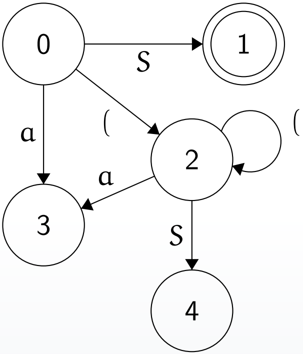
$$S' \to S \quad S \to a \quad S \to (S)$$



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

Автомат нередуцируемых префиксов

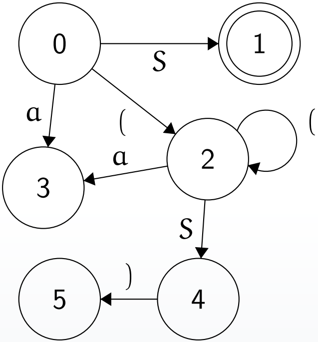
$$S' \to S \quad S \to a \quad S \to (S)$$



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

Автомат нередуцируемых префиксов

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



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

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?

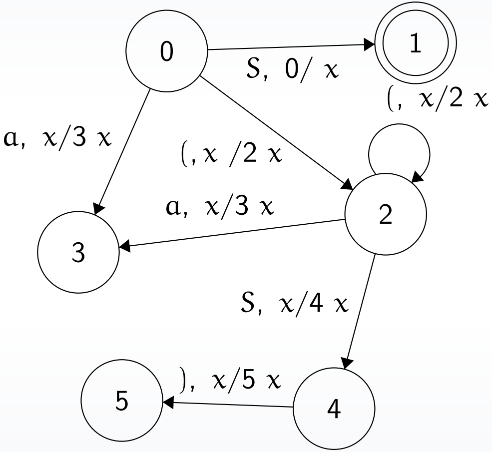
- Хранить сами сентенциальные формы плохо — проблема с извлечением нескольких подряд символов.
- Логично хранить последовательности последних символов с.ф., которые могут привести к разным свёрткам, закодированными одним символом стека.
- А это — в точности состояния автомата.

PDA по LR(0)-автомату

Общая конструкция

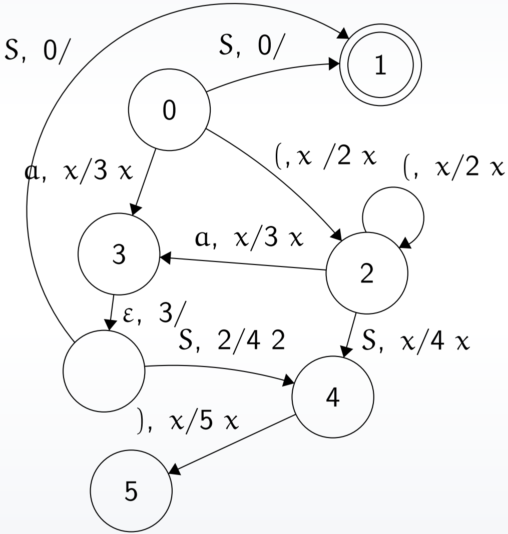
- При каждом сдвиге кладём в стек номер состояния, в которое приходим в конечном автомате.
- При каждой свёртке извлекаем из стека n символов, где n — длина правой части β правила $A \to \beta$, после чего переходим в состояние с номером $n+1$-ого символа в стеке, подразумевая на ленте символ A.
- Совершаем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

Пример построения PDA



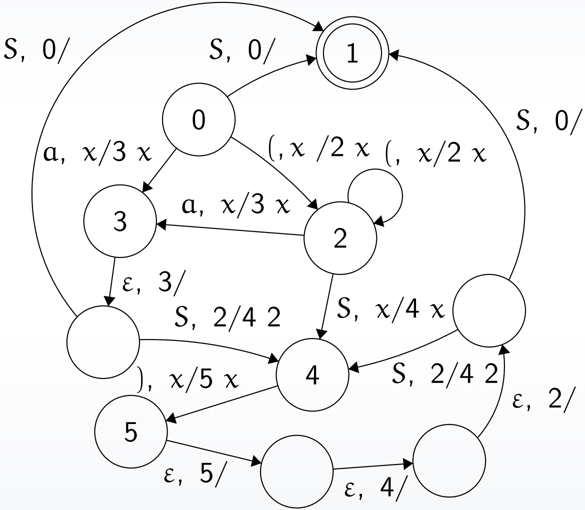
0	$S' \to \bullet S$ $S \to \bullet(S)$ $S \to \bullet a$	
1	$S' \to S\bullet$	$S \to S'$
2	$S \to (\bullet S)$ $S \to \bullet(S)$ $S \to \bullet a$	
3	$S \to a\bullet$	$a \to S$
4	$S \to (S\bullet)$	
5	$S \to (S)\bullet$	$(S) \to S$

Пример построения PDA



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

Пример построения PDA

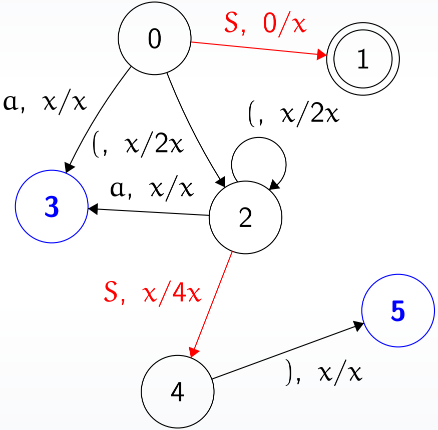


0	$S' \to \bullet S$	
	$S \to \bullet(S)$	
	$S \to \bullet a$	
1	$S' \to S\bullet$	$S \to S'$
2	$S \to (\bullet S)$	
	$S \to \bullet(S)$	
	$S \to \bullet a$	
3	$S \to a\bullet$	$a \to S$
4	$S \to (S\bullet)$	
5	$S \to (S)\bullet$	$(S) \to S$

Промежуточный PDA-распознаватель

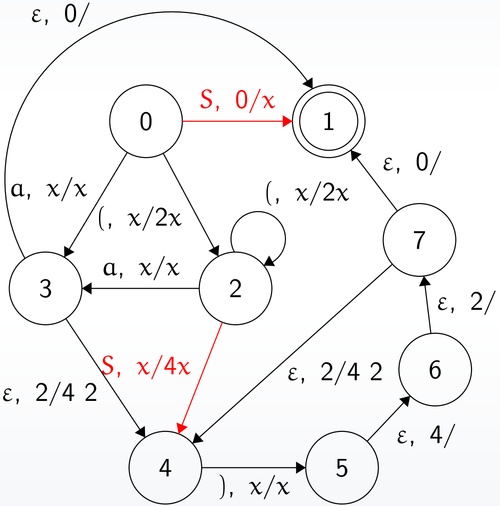
- Стековые символы, ведущие в состояния свёртки, не являющиеся финальными (у нас это 3 и 5), бесполезны, потому что сразу же безальтернативно извлекаются из стека.
- Распознаватель ещё не может быть использован как парсер, потому что он «читает> > нетерминалы с ленты. Этого можно избежать, если принять, 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 нетерми-налам



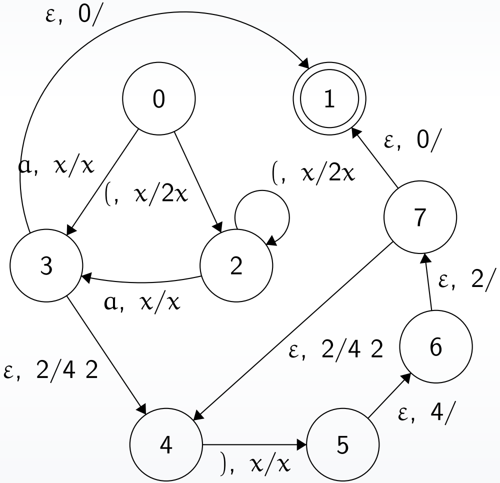
0	$S' \to \bullet S$ $S \to \bullet(S)$ $S \to \bullet a$	
1	$S' \to S\bullet$	$S \to S'$
2	$S \to (\bullet S)$ $S \to \bullet(S)$ $S \to \bullet a$	
3	$S \to a\bullet$	$a \to S$
4	$S \to (S\bullet)$	
5	$S \to (S)\bullet$	$(S) \to S$

Избавление от переходов по нетерминалам



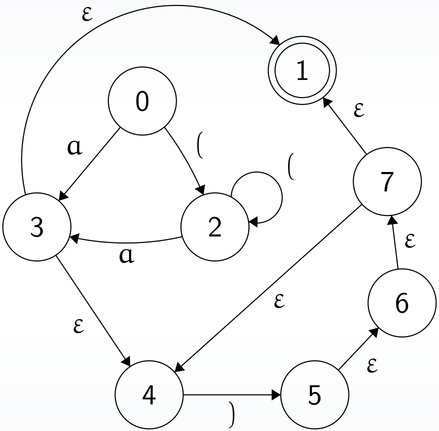
0	$S' \to \bullet S$ $S \to \bullet(S)$ $S \to \bullet a$	
1	$S' \to S\bullet$	$S \to S'$
2	$S \to (\bullet S)$ $S \to \bullet(S)$ $S \to \bullet a$	
3	$S \to a\bullet$	$a \to S$
4	$S \to (S\bullet)$	
5	$S \to (S)\bullet$	$(S) \to S$

Избавление от переходов по нетерминалам



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

Бонус — регулярная аппроксимация



Аппроксимацией исходного языка $(^n a)^n$, 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 $\beta \to A$ заглянуть в множество $\mathrm{FOLLOW}(A)$, чтобы понять, какую свёртку делать (и делать ли).

Отказ от эндмаркера и SLR

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

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

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

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 $\mathrm{SLR}(1)$-грамматика.

Следует из теоремы:

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

LR(k)-распознаватели

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

$\mathrm{LR}(k)$-распознаватели

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

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

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

LR(k)-распознаватели

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

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

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

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

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

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

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

$$S' \to S \qquad S \to bSS \qquad S \to a$$
$$S \to aac$$

Не $LR(1)$, конфликт свёртки на префиксе ba с контекстом a.
Используем трансформацию уточнения правого контекста:

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

Теперь присоединим правые контексты:

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

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:

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

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

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.

$$
\boxed{
\begin{array}{l}
S' \rightarrow \bullet S \\
S \rightarrow \bullet aSb \\
S \rightarrow \bullet aCa \\
S \rightarrow \bullet bCb \\
S \rightarrow \bullet c
\end{array}
}
$$

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

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

$$\boxed{\begin{array}{l} S' \to \bullet S \\ S \to \bullet aSb \\ S \to \bullet aCa \\ S \to \bullet bCb \\ S \to \bullet c \end{array}} \xrightarrow{a} \boxed{\begin{array}{l} S \to a \bullet Sb \\ S \to a \bullet Ca \\ S \to \bullet aSb \\ S \to \bullet aCa \\ S \to \bullet bCb \\ S \to \bullet c \\ C \to \bullet aCa \\ C \to \bullet bCb \\ C \to \bullet c \end{array}}$$

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

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

$$\boxed{\begin{array}{l} S' \to \bullet S \\ S \to \bullet aSb \\ S \to \bullet aCa \\ S \to \bullet bCb \\ S \to \bullet c \end{array}} \xrightarrow{a} \boxed{\begin{array}{l} S \to a \bullet Sb \\ S \to a \bullet Ca \\ S \to \bullet aSb \\ S \to \bullet aCa \\ S \to \bullet bCb \\ S \to \bullet c \\ C \to \bullet aCa \\ C \to \bullet bCb \\ C \to \bullet c \end{array}} \xrightarrow{c} \boxed{\begin{array}{l} S \to c\bullet \\ C \to c\bullet \end{array}}$$

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: $$\begin{aligned} S &\to aSb \mid aCa \mid bCb \mid c \\ C &\to aCa \mid bCb \mid c \end{aligned}$$

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

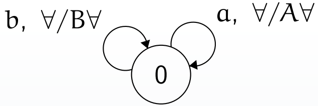
$$\begin{aligned} S &\to a[Sb] \mid a[Ca] \mid b[Cb] \mid c \\ [Sb] &\to a[Sb]b \mid a[Ca]b \mid b[Cb]b \mid cb \\ [Ca] &\to a[Ca]a \mid b[Cb]a \mid ca \\ [Cb] &\to a[Ca]b \mid b[Cb]b \mid cb \end{aligned}$$

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 c 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.

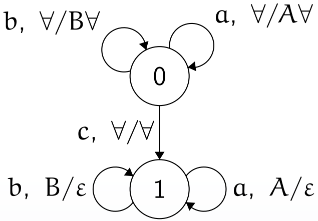


Для экономии места символ $\forall$ использован в роли параметра, пробегающего значения A, B и Z_0: на детерминированность это не влияет, поскольку переходы с его участием делаются по разным терминалам.

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.

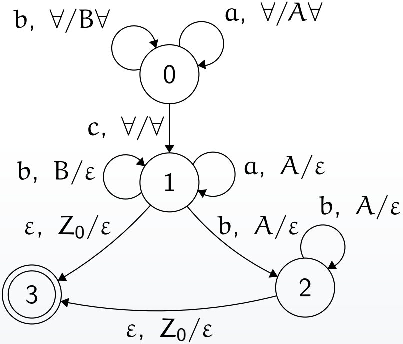


Единственная тонкость — это переход от чтения w^R к чтению b^n. Он происходит, если на вершине стека лежит A, а читается буква b, и это не приводит к неопределённости, поскольку при чтении буквы b из палиндромной части мы обязаны всегда иметь на вершине стека символ B.

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 $|x_2x_3| \leqslant p$, $|x_2| > 0$, и $\forall i(x_1x_2^ix_3y_1y_2^iy_3 \in L \; \& \; x_1x_2^ix_3z_1z_2^iz_3 \in L)$.

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 $w = xy \mathbin{\&} w' = xz$, $|x| > p$ 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 $|x_2x_3| \leqslant p$, $|x_2| > 0$, и $\forall i(x_1x_2^ix_3y_1y_2^iy_3 \in L \mathbin{\&} x_1x_2^ix_3z_1z_2^iz_3 \in L)$.

Рассмотрим язык $\{a^nb^n\} \cup \{a^nb^{2n}\}$, положим $x = a^nb^{n-1}$, $y = b$, $z = b^{2n-1}$, 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 xy, 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 xy 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

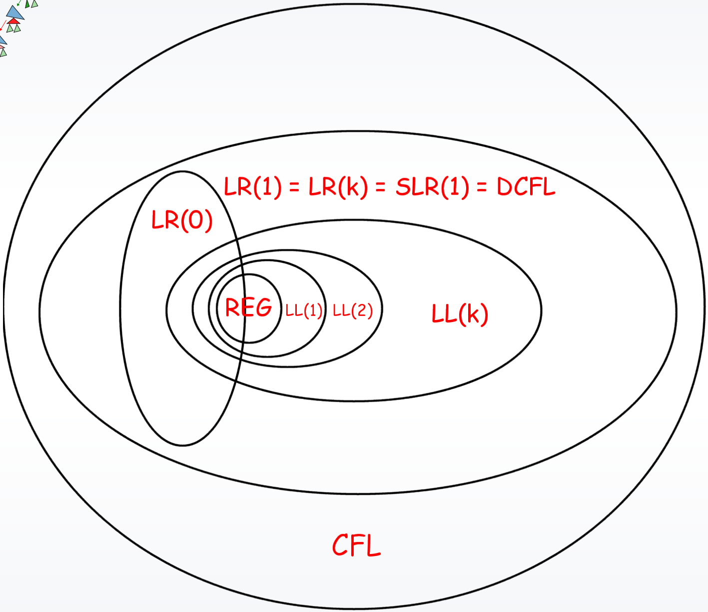
Исследовать LL(k)-свойства уже известного нам DCFL $L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}$.

- $xz = a^{n+k}ca^{n+k}$, $yz = a^{n+k}cb^{n+k}$, 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 $a^{n+k}ca^{n+k}$ при чтении символа T_z анализатору будет видно $k_1 \leqslant k$ букв a (если $k_1 < k$, то за ними будет конец слова), и начиная с этого состояния анализатор распознает суффикс a^i, $i \geqslant k_1$. В слове $a^{n+k}cb^{n+k}$ при чтении символа T_z анализатор увидит $k_2 \leqslant k$ букв b и распознает суффикс b^j, $j \geqslant k_2$.
- Если заменить в слове $a^{n+k}cb^{n+k}$ суффикс b^j на a^i, то анализатор прочитает 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 $a^{n+k}ca^{n+k}$, и анализатор определит, что слово $a^{n+k}cb^{n+k-j}a^i \in L$, что неверно. Значит, L — не LL(k).

Иерархия Хомского revisited

Утверждения ниже касаются только языков (не грамматик)!

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


LR(1) = LR(k) = SLR(1) = DCFL
LR(0)
REG
LL(1)
LL(2)
LL(k)
CFL