

Трансформации и AST. Обработка ошибок. Проблема соответствия Поста



Теория формальных языков
2022 г.



$LR(k) \rightarrow LR(1)$, 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]$		



Присоединение правого контекста

- Пусть нужно присоединить правые контексты к нетерминалу A . Для всех правил вида $C \rightarrow \gamma_1 A t \gamma_2$, где t — терминал, порождает нетерминал $[At]$ и заменяем им часть At данного правила.
- Для всех правил вида $A \rightarrow \delta$ добавляем правило $[At] \rightarrow \delta t$.
- Данное преобразование не может быть применено к правилу вида $C \rightarrow \gamma_1 A B \gamma_2$. Поэтому, если нужно присоединять контекст в таком правиле, необходимо воспользоваться алгоритмом уточнения правого контекста.



Уточнение правого контекста

- Пусть нужно уточнить правый контекст у A по правилу $C \rightarrow \gamma_1 A B \gamma_2$. Положим, что $\text{FIRST}(B)$ не содержит ϵ .
- Для каждого элемента $c \in \text{FIRST}(B)$ строим нетерминал $[c/B]$ и правило $C \rightarrow \gamma_1 A c [c/B] \gamma_2$.
- Для всех правил вида $B \rightarrow c \delta$ строим правила $[c/B] \rightarrow \delta$.
- Для всех правил вида $B \rightarrow D \delta$ таких, что $c \in \text{FIRST}(D)$, строим правила вида $[c/B] \rightarrow [c/D] \delta$. Рекурсивно замыкаем процедуру (до неподвижной точки).
- Если нужно уточнить контекст A по правилу $C \rightarrow \Phi A$, тогда ищем все правила $C' \rightarrow \Psi_1 C \Psi_2$, которые порождают C , получаем правила $C' \rightarrow \Psi_1 \Phi A \Psi_2$ и действуем с ними так же, как при обычном уточнении правого контекста.
- В полученной грамматике могут появиться ϵ -правила (кодировку для ϵ можно выбрать произвольно). Поэтому их придётся в дальнейшем устранить.



Применение MLS-подгонки

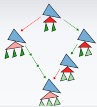
Исследовать на детерминированность язык
 $L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$

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:

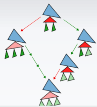
$$\begin{array}{ll} S & \rightarrow aSb \mid aCa \mid bCb \mid c \\ C & \rightarrow aCa \mid bCb \mid c \end{array}$$

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

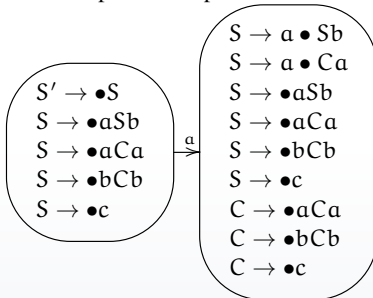
Исследовать на детерминированность язык

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

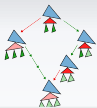
Пробная грамматика для L :

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

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

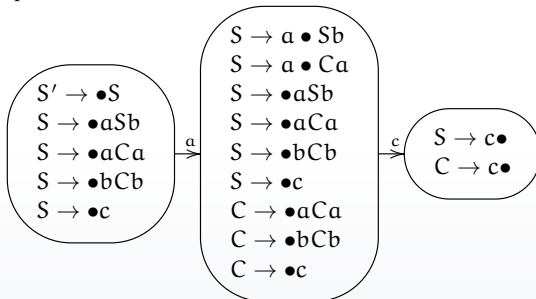
$$L = \{a^n w c w^R b^n \mid w \in \{a, b\}^*\}.$$

Пробная грамматика для L :

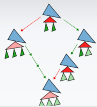
$$\begin{array}{ll} S & \rightarrow aSb \mid aCa \mid bCb \mid c \\ C & \rightarrow aCa \mid bCb \mid c \end{array}$$

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):

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

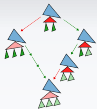
Можно построить LR(0)-автомат для этой грамматики и убедиться, что он не содержит конфликтов. Значит язык L — детерминированный (более того, LR(0)).



LL-подгонка

- Устранение левой рекурсии;
- Извлечение левого контекста:

Если даны $A \rightarrow \Phi\gamma_1$, $A \rightarrow \Phi\gamma_2$, тогда можно построить эквивалентные правила $A \rightarrow \Phi A'$, $A' \rightarrow \gamma_1 \mid \gamma_2$.



Абстрактное синтаксическое дерево

Переход от конкретного дерева разбора к дереву разбора, содержащему только значащие нетерминалы, называется переходом к AST.

- Можно сливать транзитные узлы;
- Можно стирать ветви дерева разбора.

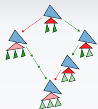
При применении подгонок и упрощений дерево разбора тоже меняется:

- устранение ϵ -правил — добавление новой абстрактной структуры;
- извлечение левого контекста — слияние сиблингов;
- присоединение и извлечение правого контекста — зависит от лексера и синхронизирующих токенов;
- устранение левой рекурсии — полностью перестраивает структуру дерева



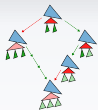
Подходы к восстановлению после ошибок в PDA

- Множество к.э. по Майхиллу–Нероуду бесконечно \Rightarrow синхронизация учитывает стек.



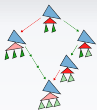
Подходы к восстановлению после ошибок в PDA

- Множество к.э. по Майхиллу–Нероуду бесконечно \Rightarrow синхронизация учитывает стек.
- Стандартный подход: множество синхронизирующих терминалов строится для каждого нетерминала отдельно.



Подходы к восстановлению после ошибок в PDA

- Множество к.э. по Майхиллу–Нероуду бесконечно \Rightarrow синхронизация учитывает стек.
- Стандартный подход: множество синхронизирующих терминалов строится для каждого нетерминала отдельно.
- (режим паники) При восстановлении после ошибки отбрасывается не только префикс ошибочного входа, но и вершина стека.



Подходы к восстановлению после ошибок в PDA

- Множество к.э. по Майхиллу–Нероуду бесконечно \Rightarrow синхронизация учитывает стек.
- Стандартный подход: множество синхронизирующих терминалов строится для каждого нетерминала отдельно.
- (режим паники) При восстановлении после ошибки отбрасывается не только префикс ошибочного входа, но и вершина стека.
- (режим починки) При восстановлении после ошибки стек не отбрасывается, а вход подгоняется под стек. Набор действий может зависеть от ячейки таблицы, содержащей ошибку.

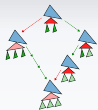


Panic mode для LL-разбора

Ошибочная ситуация

Терминал в стеке не совпадает с терминалом на ленте, либо переход по таблице правил приводит к ошибке.

- Отбрасываем вершину стека до синхронизирующего токена и входные символы до успеха перехода по нему.
- Возможное удаление \Rightarrow для токена A синхронизирующими могут предполагаться элементы $\text{FOLLOW}(A)$.
- Возможная вставка \Rightarrow синхронизирующие — $\text{FIRST}(A)$. Если конфликт терминалов — интерпретируем как возможную вставку.



Panic mode для LR-разбора

Ошибочная ситуация

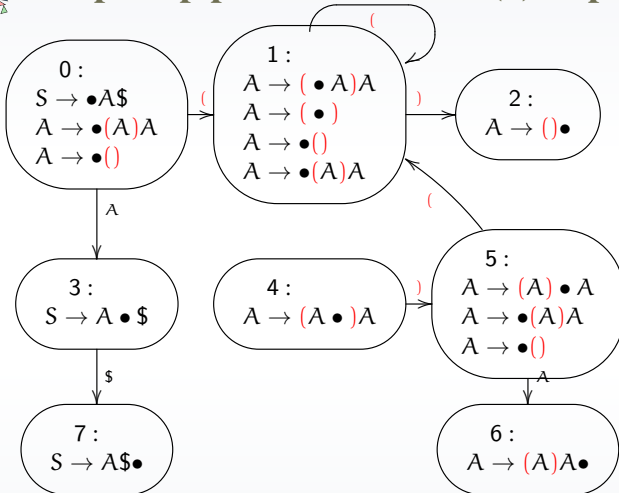
Переход по таблице правил приводит к ошибке.

- Вводим специальный токен «ошибка» в правиле $A \rightarrow \beta \bullet \alpha$, на котором она произошла.
- Отбрасываем вершину стека до свёртки по правилу $A \rightarrow \text{«ошибка»} \alpha \bullet$, не добавляя ничего в стек (если есть lookahead, то до совпадения с lookahead-ом).
Продолжаем разбор дальше.

Альтернатива: поиск «починки» — минимального количества действий, позволяющего возобновить парсинг.

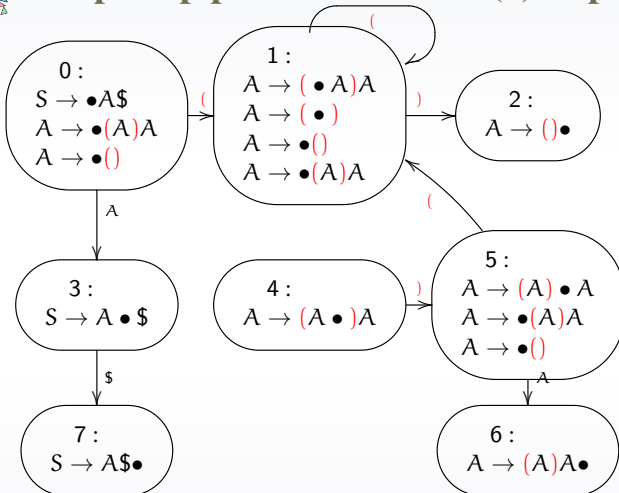


Пример panic mode в LR(0)-парсере





Пример panic mode в LR(0)-парсере



Разбор строки $()() \$$: $([0], ()() \$) \rightarrow ([1, 0], ()() \$) \rightarrow ([2, 1, 0], () \$) \rightarrow ([3, 0], () \$)$

На этом шаге происходит ошибка. Строим $S \rightarrow \bullet \text{«ошибка»} \$$, отбрасываем $()$ и редуцируемся в S .



Бурке–Фишер и его вариации

Идея алгоритма

При заранее заданном k и ошибке на i -ом терминале входа рассмотреть возможные последовательности терминалов от i -ого до $i + k - 1$ -ого, продолжающие парсинг, и выбрать в качестве «починки» ту из них, расстояние Левенштейна до которой от реального входа наименьшее.

- (Corchuello et al) Также разрешается делать операции сдвига по lookahead-у.
- (Diekmann et al) Ищутся все возможные варианты «починки» и выбирается тот из них, который позволяет продолжить разбор на наибольшую глубину.



Разрешимость проблем в КС-грамматиках

Разрешимые проблемы

- Пустота языка
- Вхождение слова в язык
- Бесконечность языка

Неразрешимые проблемы

...большинство остальных.

Подход к доказательству неразрешимости: машины Тьюринга «с историей».



«История» вычислений

Плоская конфигурация машины Тьюринга — это $P_1 q_i p_j P_2$, где P_1 — это лента слева от головки МТ, q_i — состояние МТ, p_j — ячейка ленты, на которой стоит головка, P_2 — лента справа от головки.

Тогда шаг МТ описывается как SRS на конфигурациях.

- Пусть в состоянии q_i , прочитав символ p_i , МТ записывает p'_i и сдвигает головку вправо, переходя в состояние q_j . Тогда правило переписывания имеет вид $q_i p_i \rightarrow p'_i q_j$.
- Пусть в состоянии q_i , прочитав символ p_i , МТ записывает p'_i и сдвигает головку влево, переходя в состояние q_j , причём слева от ячейки стоит символ p_{i-1} . Тогда правило переписывания имеет вид $p_{i-1} q_i p_i \rightarrow q_j p_{i-1} p'_i$.

История вычисления МТ — это $w_0 \# w_1 \# \dots \# w_F$, где w_0 — стартовая, w_F — финальная конфигурации, и w_{i+1} получается из w_i применением SRS, описывающей шаги МТ. ▲



Пересечение КС-грамматик

Рассмотрим следующие истории:

$$w_0 \# w_1^R \# w_2 \# w_3^R \dots \# w_F^G$$

Где w_i — конфигурации, начиная со стартовой и кончая какой-нибудь финальной (но не обязательно согласующиеся с правилами МТ); $w_{2.i}^G$ — это просто $w_{2.i}$; $w_{2.i+1}^G$ — это $w_{2.i}^R$ (реверсированная конфигурация).

- Язык $\mathcal{L}_1 = \{w_0 \# w_1^R \# w_2 \# w_3^R \dots \# w_F^G \mid \#w_{2.i} \text{ согласована с } \#w_{2.i+1} \text{ относительно правил МТ}\}$ является КС (достаточно попарно разобрать конфигурации как слова, получающиеся из палиндромов конечным числом правил).
- Язык $\mathcal{L}_2 = \{w_0 \# w_1^R \# w_2 \# w_3^R \dots \# w_F^G \mid \#w_{2.i+1} \text{ согласована с } \#w_{2.i+2} \text{ относительно правил МТ}\}$ является КС (аналогично).



Пересечение КС-грамматик

- Язык $\mathcal{L}_1 = \{w_0 \# w_1^R \# w_2 \# w_3^R \dots \# w_F^G \mid \# w_{2 \cdot i} \text{ согласована с } \# w_{2 \cdot i + 1} \text{ относительно правил МТ}\}$ является КС (достаточно попарно разобрать конфигурации как слова, получающиеся из палиндромов конечным числом правил).
- Язык $\mathcal{L}_2 = \{w_0 \# w_1^R \# w_2 \# w_3^R \dots \# w_F^G \mid \# w_{2 \cdot i + 1} \text{ согласована с } \# w_{2 \cdot i + 2} \text{ относительно правил МТ}\}$ является КС (аналогично).
- $\mathcal{L}_1 \cap \mathcal{L}_2 \neq \emptyset \Leftrightarrow$ язык, порождаемый МТ, не пуст.
- Следовательно, проблема $\mathcal{L}_1 \cap \mathcal{L}_2 \stackrel{?}{=} \emptyset$ неразрешима для КС-языков.



Неразрешимость всеобщности

Язык $\mathcal{L}_{\text{fail}} = \Sigma^* \setminus (\mathcal{L}_1 \cap \mathcal{L}_2)$ является КС. КС-грамматика — объединение грамматики для языка, где хотя бы один переход с чётного шага истории на нечётный не согласуется с правилами МТ, грамматики для языка, где несогласование есть при переходе с нечётного шага на чётный, и грамматики для языка слов, имеющих неправильную лексическую структуру.

Следствие

Вопрос $\mathcal{L} \stackrel{?}{=} \Sigma^*$ неразрешим для КС-грамматик (т.к. если есть способ разрешать $\mathcal{L}_{\text{fail}} \stackrel{?}{=} \Sigma^*$, тогда есть способ и разрешить проблему пустоты языка МТ).



Проблема соответствия Поста

Рассмотрим «домино» из пар $\langle u, w \rangle \in \langle \Sigma^*, \Sigma^* \rangle$. Пусть имеется n таких пар вида $\langle u_i, w_i \rangle$. Существует ли последовательность индексов, такая что

$$u_{i_1} \dots u_{i_k} = w_{i_1} \dots w_{i_k}?$$

- Неразрешима — рассмотрим пошаговые «истории» МТ.
- Следствие — вопрос о неоднозначности КС-грамматики тоже неразрешим; вопрос о вхождении палиндрома в КС-язык неразрешим; вопрос о вхождении квадрата в КС-язык неразрешим.



Теорема Грейбах

Пусть C — семейство языков, содержащее все регулярные языки, для которого неразрешима проблема всеобщности. Если это семейство замкнуто относительно объединения и приписывания регулярных языков, то для него неразрешимо никакое свойство, выполняющееся для всех регулярных языков и замкнутое относительно производных.