18. Анализ на основе LR(0)-автомата. SLR(1)-анализ.

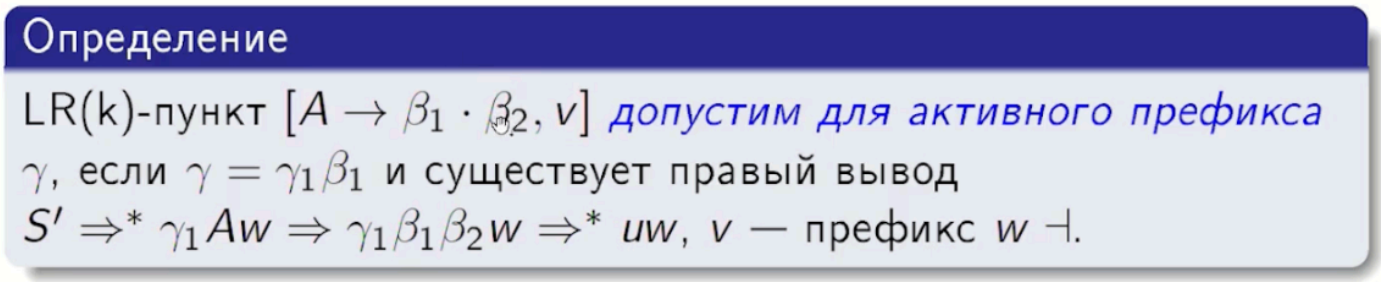
Анализ на основе LR(0)-автомата и его расширение — SLR(1)-анализ — представляют собой методы **восходящего синтаксического анализа** типа «перенос–свёртка».
Эти методы используют детерминированный конечный автомат для отслеживания **активных префиксов** грамматики в стеке.

1. LR(0)-пункты и автомат

Основой LR-анализа является понятие **LR(0)-пункта**.

опр Пункт — это правило вывода с точкой в правой части, например $[A \to \beta_1 \cdot \beta_2]$.

А если быть точнее:
k=0



- **Точка** обозначает границу между тем, что уже находится в стеке (β_1), и тем, что ожидается во входном потоке (β_2).
- Если точка стоит в конце ($[A \to \alpha\cdot]$), это означает, что в стеке находится **основа** и можно выполнить **свёртку**.

В LR(0) автомате мы можем принимать решение в текущий момент времени и не заглядывая ни на один символ ленты вперед.

LR(0)-автомат (A_G) — это детерминированный конечный автомат, состояниями которого являются множества допустимых пунктов.

Хз нужно ли следующие следствия рассказывать:

Следствие

Множество пунктов, достежимых из $\mathcal{I}_0$ по пути помеченного γ, совпадают с множеством допустимых пунктов для активного префикса γ.

Следствие

Если в состоянии A_G есть пункты $[A \to \alpha_1 X \alpha_2]$ и $[B \to \beta_1 Y \beta_2]$, то X=Y.

Для построения автомата $\mathcal{A}_G$ не нужно вначале строить автомат пунктов (он неявно задан самой грамматикой). Детерминированный LR(0)-автомат можно получить сразу, используя две вспомогательные функции. Пусть M — произвольное множество пунктов грамматики.

Определение

CLOSURE(M) — минимальное по включению множество пунктов, содержащее M, и такое, что вместе с пунктом вида $[A \to \alpha \cdot B\beta]$ в нём содержатся все пункты вида $[B \to \cdot\gamma]$.
$\text{CLOSURE}(M) = \{i | \exists j \in M \wedge \exists (i,j) - \text{путь, помеченный } \lambda\}$.

Множество CLOSURE(M) – это и есть замыкание в терминах λ-НКА.

Чтобы построить CLOSURE(M), в качестве первого приближения берём множество M и упорядочиваем его пункты в очередь. В цикле обрабатываем первый элемент в очереди, добавляя пункты вида $[B \to \cdot\gamma]$ в CLOSURE(M) и конец очереди, если их ещё не было в CLOSURE(M). Первый элемент из очереди всегда удаляется. Когда очередь становится пустой, построение CLOSURE(M) завершено.

Определение

Пусть X — произвольный символ грамматики. Тогда
$\text{GOTO}(M, X) = \text{CLOSURE}(\{[A \to \alpha X \cdot \beta] | [A \to \alpha \cdot X\beta] \in M\})$.

Функция GOTO является функцией переходов LR(0)-автомата $\mathcal{A}_G$.

Алгоритм построения LR(0)-автомата

Вход: Расширенная грамматика $G = (\Sigma, \Gamma, P, S')$.
Выход: ДКА $\mathcal{A}_G = (Q, \Sigma \cup \Gamma, \delta, I_0, Q)$.

1 $I_0 = \text{CLOSURE}([S' \to \cdot S])$
2 $label(I_0) = 0$
3 $Q = \{I_0\}$
4 цикл пока $\exists I \in Q : label(I) = 0$
5 цикл по $X \in \Sigma \cup \Gamma$
6 $\delta(I, X) = \text{GOTO}(I, X)$
7 если $\delta(I, X) \notin Q$
8 $Q = Q \cup \{\delta(I, X)\}$
9 $label(\delta(I, X)) = 0$
10 $label(I) = 1$

Анализ на основе LR(0)-автомата

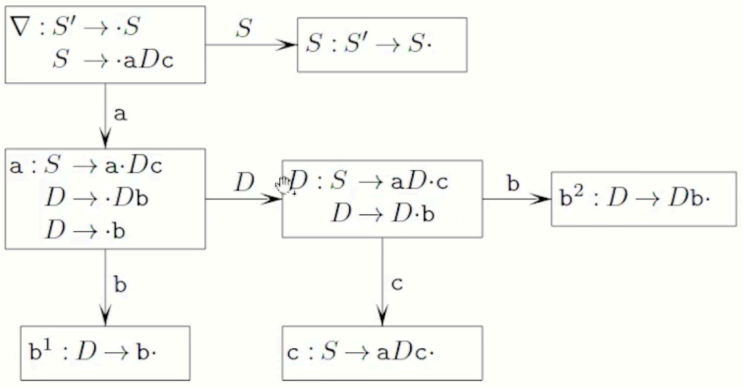
Начнём с неформального описания роли LR(0)-автомата в анализе «перенос–свертка». Подадим на вход автомата $\mathcal{A}_G$ цепочку w, тестируемую на выводимость в грамматике G. После каждого такта работы автомата будем складывать новое состояние автомата в стек(а значит, и прочитанный символ) и анализировать это новое состояние, возможно, производя некоторые манипуляции перед тем, как разрешить автомату следующий такт.

Построение таблицы LR(0)-анализатора

Вход: Расширенная грамматика $G = (\Sigma, \Gamma, P, S')$.
Выход: Таблицы ACTION и GOTO для LR-анализа грамматики G.

```
1  построить LR(0)-автомат A_G
2  проиндексировать строки таблиц ACTION и GOTO
   состояниями A_G
3  цикл по всем состояниям I
4      если (I = [S' → S·]) ACTION(I, ⊣) = ✓
5      иначе если (I = [A → α·])
6          для каждого a ∈ Σ ∪ {⊣}
7              ACTION(I, a) = number(A → α)
8      иначе для каждого a ∈ Σ
9              если (δ(I, a) ≠ ∅) ACTION(I, a) = δ(I, a)
10         для каждого A ∈ Γ
11             если (δ(I, A) ≠ ∅) GOTO(I, A) = δ(I, A)
```

Грамматика $S' \to S, \quad S \to aDc, \quad D \to Db^1 \mid b^2$.



	ACTION				GOTO	
	a	b	c	$\dashv$	S	D
S				✓		
D		$\leftarrow b^2$	$\leftarrow c$			
a		$\leftarrow b^1$				D
b^1	3	3	3	3		
b^2	2	2	2	2		
c	1	1	1	1		
∇	$\leftarrow a$				S	

SLR(1)-анализ (Simple LR)

Большинство реальных грамматик (например, грамматики арифметических выражений) не являются LR(0)-грамматиками, так как их состояния часто содержат **конфликты**,:

- **Перенос–свёртка:** в одном состоянии есть и $[A \to \alpha\cdot]$, и $[B \to \beta \cdot a\gamma]$.
- **Свёртка–свёртка:** в состоянии два пункта вида $[A \to \alpha\cdot]$ и $[B \to \beta\cdot]$.

При заполнении строки ACTION для состояния, содержащего пункт вида $A \to \alpha\cdot$, *свертку достаточно записать в клетках, проиндексированных символами из* FOLLOW(A).
Изменив построение таблицы ACTION указанным способом, мы получаем *SLR(1)-анализатор* (простой LR(1)-анализатор).
Грамматики, для которых построенная таким способом таблица ACTION не содержит множественных записей (т. е. конфликтов), называются *SLR(1)-грамматиками*.

-
-

SLR(1)-анализ расширяет возможности LR(0) за счёт использования множеств **FOLLOW** (символов, которые могут следовать за нетерминалом в корректных формах).

- **Правило свёртки в SLR(1):** свёртка по правилу $A \to \alpha$ выполняется только в том случае, если очередной символ входной цепочки **принадлежит множеству** $FOLLOW(A)$.
- Если этот символ не входит в $FOLLOW(A)$, свёртка считается некорректной. Это позволяет разрешить многие конфликты, где LR(0)-анализатор не мог сделать выбор.