> *04.09.19*

LR-анализ

> Сегодня посмотрим только на готовый анализатор. Как его создавать посмотрим позднее.
>
> Самая важная тема курса!

Готовый анализатор — автомат с магазинной памятью. Главное отличие автомата от обычного — возможность вынуть более одного символа из стека за раз. Но заглянуть в стек больше, чем на один символ, по-прежнему нельзя! Но стековый символ может нести в себе информацию о символах ниже.

> Заучивать не надо — КСГ распознаются МПА.

С какими грамматиками будем работать:

- однозначными, так как нам нужен детерминированный алгоритм
- приведённые

Можем работать с аннулирующими правилами!

Будем рассматривать LR(k) грамматики.

Вспомним нашу арифметическую грамматику.

$E \to E \overset{(1)}{+} T | \overset{(2)}{T}$

$T \to T \overset{(1)}{*} F | \overset{(4)}{F}$

$F \to \overset{(5)}{(}E) | \overset{(6)}{x}$

в GOTO — нетерминалы грамматики. Она говорит о том, что нужно делать во время свёртки

Без цифр — перенос. С цифрой — свёртка по правилу из грамматики.

	A	C	T	I	O	N	GO		TO
	+	*	x	(	)	$\dashv$	E	T	F
E^1	$+$					✓			
T^1	(2)	$*$			(2)	(2)			
F^1	(4)	(4)			(4)	(4)			
$($			x	(			E^2	T^1	F^1
x	(6)	(6)			(6)	(6)			
$+$			x	(				T^2	F^1
$*$			x	(					F^2
E^2	+				)				
T^2	(1)	*			(1)	(1)			
F^2	(3)	(3)			(3)	(3)			
$)$	(5)	(5)			(5)	(5)			
∇			x	(			E^1	T^1	F^1

Если происходит перенос, то вершина на стеке остаётся!

Анализатор смотрит на вершину на стеке, на символ, делает то, что написано

Пример

Вершину стека пишем справа!

$(x + x) * x$

Стек	Остаток входной строки	Комментарий
∇	$(x+x)*x\dashv$	
$\nabla($	$x+x)*x\dashv$	
$\nabla(x$	$+x)*x\dashv$	Свёртка по (6)
$\nabla(F^1$	$+x)*x\dashv$	F^1 — потому что на стеке была открытая скобка, а свернулись к F. См. GOTO
$\nabla(T^1$	$+x)*x\dashv$	Свернулись по (4)
$\nabla(E^2$	$+x)*x\dashv$	Свернулись по (2). Не было бы скобки, было бы E^1
$\nabla(E^2+$	$x)*x\dashv$	
$\nabla(E^2+x$	$)*x\dashv$	
$\nabla(E^2+F^1$	$)*x\dashv$	Свёртка по (6)
$\nabla(E^2+T^2$	$)*x\dashv$	Свёртка по (4). Взяли F, сворачиваемся к T, на стеке +
$\nabla(E^2$	$)*x\dashv$	Свёртка по (1). Его длина — 3, поэтому и из стека выкидываем 3 символа.
$\nabla(E^2)$	$*x\dashv$	
∇F^1	$*x\dashv$	
∇T^1	$*x\dashv$	
∇T^1*	$x\dashv$	
∇T^1*x	$\dashv$	
∇T^1*F^2	$\dashv$	
∇T^1	$\dashv$	
∇E^1	$\dashv$	
✓	✓	

Внешняя аксиома S'

$G=(\Sigma,\Gamma,P,S)$

$G'=(\Sigma,\Gamma\cup\{S'\},P\cup\{S'\to S\},S')$

Свёртка по добавленному правилу говорит, что свёртка произошла по первой аксиоме. S' в правых частях отсутствует, поэтому она — начало. Свёртка к ней — это команда допуска.

> *Распознаватель* — таблица, которую мы нарисовали выше. *Автомат* — другая штука, который будем рассматривать далее. Состояния этого автомата — стековый алфавит анализатора.

Автомат LR(0)-пунктов

Опр. LR(k)-**пункт** в $G = (\Sigma, \Gamma, P, S)$ это четвёрка $[A \to \beta_1 \cdot \beta_2, v]$, где

- $A \to \beta_1 \beta_2 \in P$
- $|v| = k$ или $|v| < k$ и $v[|v|] = \dashv$

Пункт — разделение правила на две части. Он указывает, какую часть продукции мы уже посмотрели в данной точке в процессе синтаксического анализа.

> Зачем это нужно? Чтобы отслеживать, насколько мы готовы к свёртке.
>
> Например, пункт $A \to \cdot XYZ$ указывает, что во входном потоке мы ожидаем встретить строку, порождаемую XYZ. Пункт $A \to X \cdot YZ$ указывает, что нами уже просмотрела строка, порождаемая X , и мы ожидаем получить из входного потока строку, порождаемую YZ. Пункт $A \to XYZ \cdot$ говорит о том, что уже обнаружено тело XYZ и что, возможно, пришло время свернуть XYZ в A.

LR(0)-пункт — просто правило с точкой.

$E \to E + T$:

$E \to \cdot E + T$ $E \to E \cdot + T$ $E \to E + \cdot T$ $E \to E + T \cdot$

Теперь построим автомат пунктов, который будем использовать для принятия решений в процессе синтаксического анализа — **LR(0)-автомат**. Как этот автомат используется? Если в нём есть переход по текущему символу входной строки, то осуществляется перенос. Если нет, то произведём свёртку по правилу, о котором сообщает пункт.

Опр. **Автоматом LR(0)-пунктов** расширенной грамматики $G = (\Sigma, \Gamma, P, S')$ называется λ-НКА $I_G = (\Sigma \cup \Gamma, I, \delta, i_o, I)$, где :

- I — множество LR(0)-пунктов грамматики

  > I — начальные состояния, совпадают с конечными

- $i_0 = [S' \to \cdot S]$
- δ — множество переходов вида: $[A \to \beta_1 \cdot x\beta_2] \xrightarrow{x} [A \to \beta_1 x \cdot \beta_2]$ — базисные

  $[A \to \beta_1 \cdot B\beta_2] \xrightarrow{\lambda} [B \to \cdot\beta]$

  > Если после точки стоит терминал, то из этого пункта по этому терминалу можно перейти в следующее состояние, в котором точка передвинется на шаг вправо.
  >
  > Если после точки стоит нетерминал, то можно перейти по лямбде в следующее множество пунктов: все правила для этого нетерминала, в самом начале правых частей которых стоит точка.
  >
  > // Переход по лямбде == замыкание

Базисные пункты — $[A \to \beta_1 \cdot \beta_2]$, $\beta_1 \neq \lambda$ и i_0

> Начальный пункт и все пункты, у которых точки расположены не у левого края

Опр. **Активный префикс** — префикс r-формы, не выходящий за правый конец основы.

> == префикс, который может находится в стеке нашего анализатор.
>
> В стеке не может лежать основа и ещё что-то над ней, потому что как только мы видим основу, мы её сворачиваем

Опр. Пункт $[A \to \beta_1 \cdot \beta_2, v]$ **допустим** для активного префикса γ, если $\exists$ правый вывод $S \Rightarrow^+ \gamma' A w \Rightarrow \gamma' \beta_1 \beta_2 w \Rightarrow^* uw$, где $\gamma = \gamma' \beta_1$, v — префикс w $\dashv$

LR(0)-автомат **распознаёт** активные префиксы

Основная теорема LR-анализа

LR(0)-пункт i допустим для активного префикса $\gamma \iff$ в I_G существует пусть из i_0 в i, помеченный γ

Лемма 1

Для активного префикса γ существует допустимый для него базисный пункт.

Д-во

Давайте рассмотрим вывод, в котором появляется данный активный префикс. Нас интересует первое появление. Как только встретили форму, у которой появился такой префикс

$S \Rightarrow^* \gamma\alpha \Rightarrow^* w$

Перед появлением префикса было применено какое то правило, значит, префикс откуда-то взялся, и его можно свернуть до этого нетерминала. Если основа лежит в альфе, то предыдущая форма тоже начиналась с гаммы, а значит это не первое появление. Теперь покажем, что основа лежит на границе гаммы и альфы —$\beta_1 \beta_2$.

Если $\beta_1 \neq \lambda$, тогда $\gamma \neq \lambda$ и $S \Rightarrow^* \gamma' A \alpha; \Rightarrow \gamma' \beta_1 \beta_2 \alpha' \Rightarrow^*$, где $\gamma' \beta_1 = \gamma$, $\beta_2 \alpha' = \alpha$ и пункт $[A \to \beta_1 \cdot \beta_2]$ — допустим

Если $\gamma = \lambda$, то i_0 допустим для γ

■

Лемма 2

Пункт $[B \to \cdot\beta]$ допустим для активного префикса $\gamma \iff$ он достижим по λ-переходу из некоторого базисного пункта, допустимого для γ.

Д-во

$\Leftarrow$ Рассмотрим допустимый для γ пункт $[A \to \beta_1 \cdot B\beta_2]$. Надо перейти к пункту из условия. Существует правый вывод $S' \Rightarrow^* \gamma' Aw \Rightarrow \gamma'\beta_1 \cdot B\beta_2 w \Rightarrow uw$.

> Где то тут использовали первую лемму, чтобы обосновать возможность таких переходов в выводе.
>
> Мы не заботились о том, как преобразовывали формы. Но, так как грамматика приведённая, из нетерминала B можно вывести разные терминальные цепочки:
>
> $S' \Rightarrow^* \gamma' Aw \Rightarrow \gamma'\beta_1 B\beta_2 w \Rightarrow^* \gamma'\beta_1 Buw \Rightarrow \gamma'\beta_1\beta uw \Rightarrow^* vuw$

$\Rightarrow$

$[B \to \cdot\beta]$, значит, существует правый вывод: $S' \Rightarrow^* \gamma Bw \Rightarrow \gamma\beta w \Rightarrow^* uw$. По первой лемме для γ существует допустимый базисный пункт $[A \to \beta_1 \cdot \beta_2]$, который мы хотим найти.

Распишем тот же самый вывод, уточняя переходы через беты.

$S' \Rightarrow^* \gamma' Au \Rightarrow \gamma'\beta_1\beta_2 u = \gamma\beta_2 u = [\gamma Bw \Rightarrow \gamma\beta w] \Rightarrow^* \gamma Bw \Rightarrow \gamma\beta w \ldots$

1) $\gamma\beta_2 u = \gamma Bv$

$[A \to \beta_1\beta_2] \overset{\lambda}{\to} [B \to \cdot\beta]$

2) β_2 разворачивается в B\alpha

β_2начинается с нетерминала C, из которого выводится цепочка, начиная с B

> Существенно, что вывод правосторонний! Если бы это было не так, то могла бы случиться такая ситуация: $\beta \to DC$

$\gamma\beta_2 u = \gamma C\alpha_1 u \Rightarrow^* \gamma Cu_1 u \Rightarrow \gamma C_2\alpha_2 u_1 u \Rightarrow \gamma C_2 u_2 u_1 u \Rightarrow^* \ldots \gamma Bu_k \ldots u_1 u$

$A \to \beta_a C\alpha_1$

$\mathrm{C} \to C_2\alpha_2$

$C_2 \to C_3\alpha_3$

$C_{k-1} \to B\alpha_k$

Все лежат в P

$[A \to \beta_1 C\alpha_1] \overset{\lambda}{\to} [C \to \cdot C_2\alpha_2] \overset{\lambda}{\to} [C_2 \to \cdot C_3\alpha_3] \overset{\lambda}{\to} \ldots \overset{\lambda}{\to} [C_{k-1} \to \cdot B\alpha_k] \overset{\lambda}{\to} [B \to \cdot\beta]$

■

Доказательство теоремы

$\Leftarrow$

Индукция по длине γ

БИ. $|\gamma| = 0$

$i_0 = [S' \to \cdot S]$ допустима для $\gamma = \lambda$ по определению

Всё, что достижимо из i_0, по λ-переходу достижимо для $\gamma = \lambda$ по второй лемме

ШИ. Ненулевая длина, значит на конце есть какой-то символ

$\gamma = \bar{\gamma}X$

Последний базисный переход в пути, помеченном γ

$[A \to \beta_1 \cdot X\beta_2] \xrightarrow{X} [A \to \beta_1 X \cdot \beta_2]$

$[A \to \beta_1 \cdot X\beta_2]$ допустим для $\bar{\gamma}$ по ПИ $\Rightarrow \exists$ правосторонний вывод

$S' \Rightarrow^* \gamma' Aw \Rightarrow \gamma'\beta_1 X\beta_2 w \Rightarrow^* uw \Rightarrow [A \to \beta_1 \cdot X\beta_2]$ допустим для γ

$\bar{\gamma} = \gamma'\beta_1 \ \gamma = \gamma'\beta_1 X$

Все пункты, достижимые из $[A \to \beta_1 \cdot X\beta_2]$ по \lambda-переходам, допустим для \gamma по второй лемме.

$\Rightarrow$

Индукция по $|\gamma|$

БИ. $\gamma = \lambda$

i_0 допустим для γ

Воспользуемся леммой два. Все переходы по пустому слову из i_0 достижимы и допустимы.

ШИ.

$\gamma = \bar{\gamma}X$

$[A \to \beta_1 \cdot \beta_2]$ допустим для $\gamma \Rightarrow \exists$ правосторонний вывод:

$S' \Rightarrow^* \gamma' Aw \Rightarrow \gamma'\beta_1 X\beta_2 w \Rightarrow^* uw$

$\gamma = \gamma'\beta_1 = \bar{\gamma}X$

$\beta_1 = \beta_1' X$

$[A \to \beta_1 \cdot \beta_2] = [A \to \beta_1' X \cdot \beta_2]$

$[A \to \beta_1' \cdot X\beta_2]$ достижим в I_G по пути, помеченному $\bar{\gamma}$ (по ПИ)

$[A \to \beta_1 \cdot X\beta_2] \xrightarrow{X} [A \to \beta_1 X \cdot \beta_2]$

■

> *19.09.19*

LR(0)-автомат

Следствие 1. Автомат I_G распознаёт язык активных префиксов грамматики G

Автомат, который мы построили — недетерминированный из-за лямбда-переходов. Поэтому построим *эквивалентный ДКА*, **LR(0)-автомат** — A_G . Обычно он рассматривается как неполный, поэтому все состояния — терминальные.

Следствие 2. Состояние i автомата A_G, достижимое из i_0, по пути, помеченному γ, совпадает с множеством пунктов, допустимых для активного префикса γ

Следствие 3. Если в состоянии автомата A_G есть пункт $A \to \beta_1 X \cdot \beta_2$ и $B \to \beta_1' Y \cdot \beta_2'$, то $X = Y$.

Как построить состояния?

Опр. CLOSURE(M) — минимальное по включению множество пунктов, содержащее M, такое, что если в нём содержится пункт вида $[A \to \beta_1 \cdot B\beta_2]$, то в CLOSURE(M) сдержатся все пункты вида $[B \to \cdot\beta]$

> Добавляем все пункты, которые можно получить переходом по лямбде

CLOSURE(M) = $\{i | i \in M$ и $\exists (i,j)$-путь, помеченный $\gamma\} \cup M$

$i_0 = [S' \to \cdot S]$

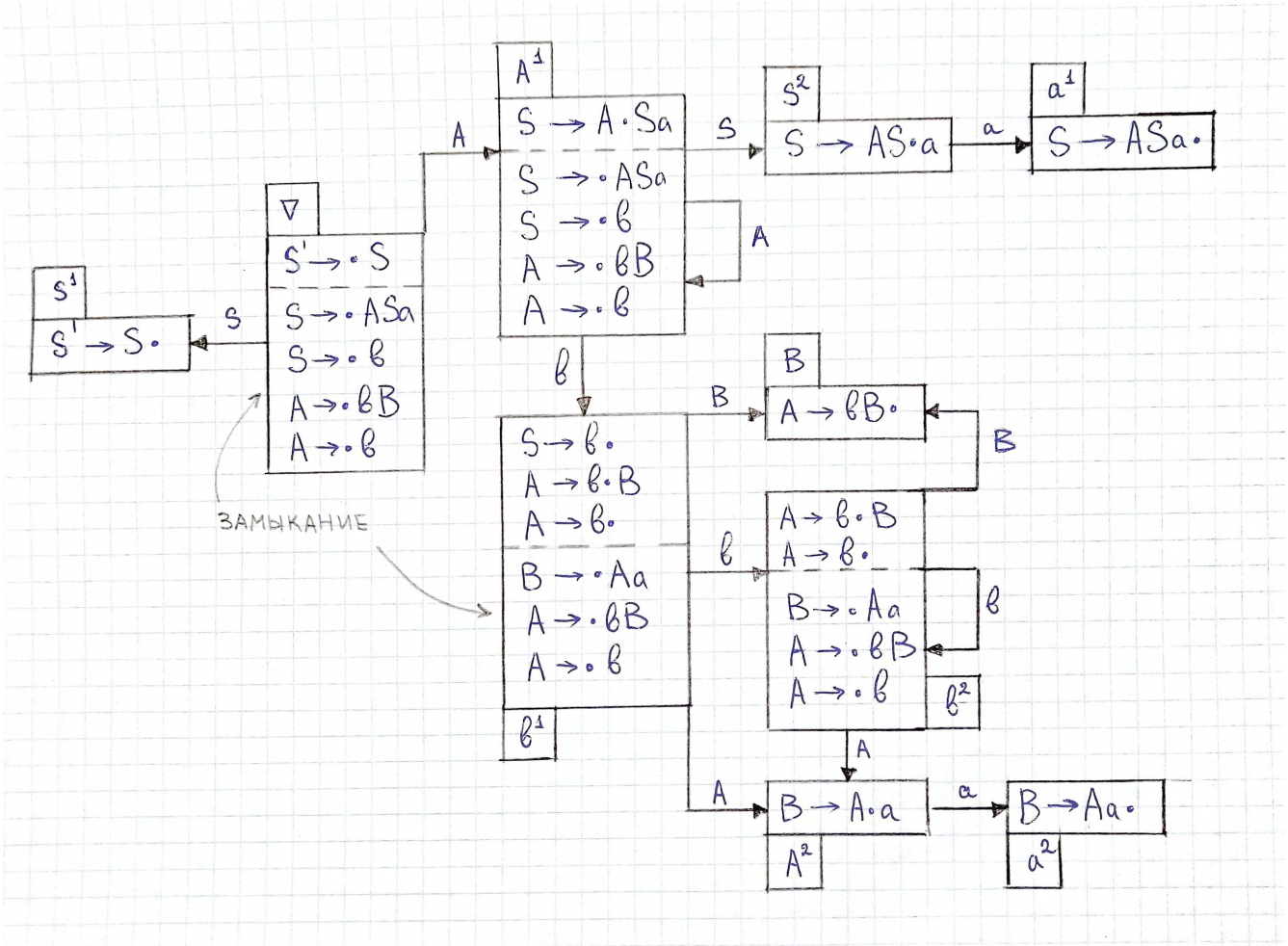
В A_G начальное состояние — CLOSURE($\{i_0\}$)

GOTO(M,X) — новые состояния, функция перехода в ДКА

GOTO(M,X) = CLOSURE($\{[A \to \beta_1 X \cdot \beta_2] | [A \to \beta_1 \cdot X\beta_2] \in M\}$)

Пример

$S' \to S$ $S \to ASa|b$ $A \to bB|b$ $B \to Aa$



Опр. LR(0)-грамматика — грамматика, LR(0)-автомат которой не содержит конфликтов. То есть любое состояние, содержащее пункт вида $[A \to \beta\cdot]$, содержит ровно 1 пункт ???

Конфликты

β и β_1 кончаются на один символ (см. следствие 3)

1. Перенос-свёртка:

   $A \to \beta\cdot$

   $B \to \beta_1 \cdot a\beta_2$

   > Если точка стоит перед нетерминалом — будем замыкать, и всё равно появится терминал. И либо первый конфликт, либо второй

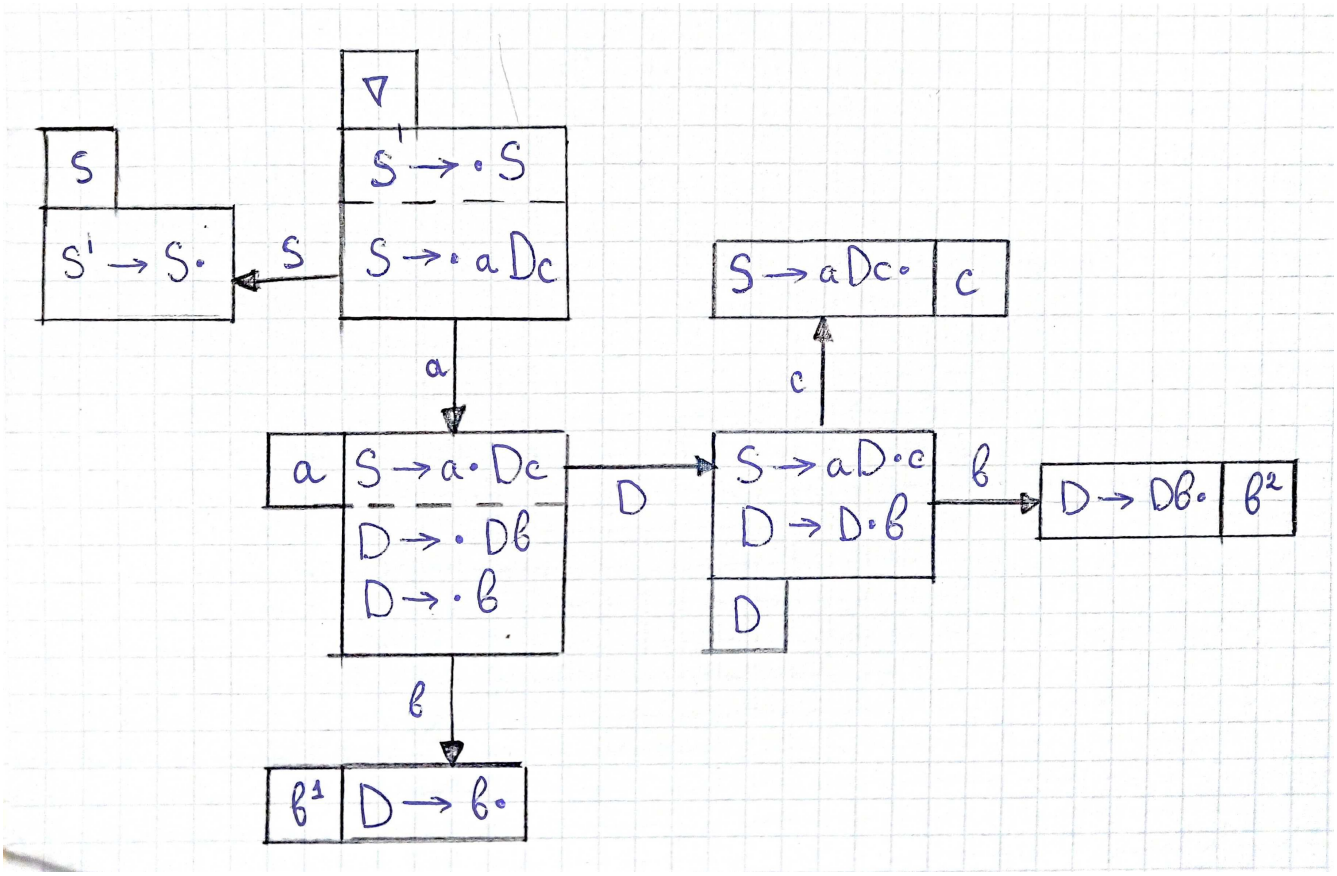
2. Свёртка-свёртка:

   $A \to \beta_1\cdot$

   $B \to \beta_2\cdot$

Другой пример

$S' \to S \; S \to aDc \; D \to Db|b$



Стек	Входная строка
∇	$abbbc \dashv$
∇a	$bbbc \dashv$
∇ab^1	$bbc \dashv$
откатываемся на длину основы (правой части)	
∇a	$bbc \dashv$
∇aD	$bbc \dashv$
∇aDb^2	$bc \dashv$
откатываемся на 2 элемента к состоянию D	
∇a	$bc \dashv$
∇aD	$bc \dashv$
∇aDb^2	$c \dashv$
∇a	$c \dashv$
∇aD	$c \dashv$
∇aDc	$\dashv$
∇	$\dashv$
∇S	$\dashv$
✓	$\dashv$

Построение LR(0)-распознавателя

Для расширенной грамматики

1. Построить A_G
2. Описать таблицы ACTION, GOTO
3. Проиндексировать строки таблицы (стековые алфавиты) состояниями A_G
4. Проиндексировать столбцы ACTION символами из $\Sigma \cup \dashv$
5. Проиндексировать столбцы GOTO символами из Γ
6. $\forall \delta(I, a) : ACTION(I, a) = \delta(I, a), a \in \Sigma$
7. $ACTION([S' \to S\cdot], \ \dashv) =$ ✓

SLR(1)-грамматики

26.09.19

Поменяем грамматику: $S' \to S$ $S \to aDb$ — b вместо c $D \to Db|b$

Тогда у нас случится конфликт.

В стеке лежит какая то обработанная часть, часть основы и символ

___β a___

___B a___

Если множества FOLLOW не пересекаются, то конфликт можно разрешить. *В чью пользу*?

$FOLLOW(S) = \{\dashv\}$

$FOLLOW(D) = \{b\}$

Сворачиваемся к тому нетерминалу, у которого во множестве FOLLOW есть данный терминал

Также можно решить конфликт перенос-свёртка.

Опр. Грамматики с бесконфликтными таблицами называются SLR(1)-грамматиками

$I = [B \to \beta\cdot] \Rightarrow ACTION(I, a) = № (B \to \beta) \ \ \forall a \in FOLLOW(B)$

ACTION

	a	b	$\dashv$	
b^2		(2)	(1)	
b^1		(3)		

$E \to E \overset{(1)}{+} T | \overset{(2)}{T}$

$T \to T \overset{(1)}{*} F | \overset{(4)}{F}$

$F \to \overset{(5)}{(}E) | \overset{(6)}{x}$

Включения: $LR(0) \subset SLR(1)$

Фото автомата 26.09. ~13:15

FOLLOW:

- $E = \{\dashv, +,)\}$
- $T = \{\dashv, +, *,)\}$
- $F = \{\dashv, +, *\}$

Если видим умножение — то переносим, потому что есть такой переход в автомате. Если видим символы, которые есть в FOLLOW нетерминала слева, то можно к нему свернуться

| ACTION |

	$+$	$*$	$($	$)$	x	$\dashv$
E^1	$+$					✓
T^1	(2)	*		(2)		(2)
T^2	(1)	*		(1)		(1)

Ещё пример

$S \to ac|bDc|Da$ $D \to a$

Фото автомата от 26.09 ~13:32

Есть конфликт в состоянии a. Либо сворачиваемся к D, либо переносим. $FOLLOW(D) = \{a, c\}$. Если на входе окажется c, то у нас конфликт, непонятно, что делать. НО. Этот пункт оказался из-за переноса замыкания в предыдущем состоянии — $D \to \cdot a$. А это замыкание было замыкания другого пункта, $S \to \cdot Da$. Значит, после D должно быть a, и если после него окажется c, то это ерунда. Значит, нужно символ переносить, но по одному FOLLOW мы это понять не можем. Вот так мы и переходим к LR(1)-грамматикам

Добавим к пунктам символы.

LR(1) анализ

Опр. LR(1)-пункт — $[A \to \beta_1 \cdot \beta_2, a]$. $\uparrow$ — **ядро**

LR(1)-пункт **допустим**, если $\exists$ правый вывод $S \Rightarrow^+ \gamma' Aw \Rightarrow \gamma' \beta_1 \beta_2 w \Rightarrow^* uw$, где:

- $\gamma = \gamma' \beta_1$ — активный префикс
- v — префикс $w \dashv$
- a — первый символ $w \dashv$

> То, с чего должна начинаться оставшаяся строка, чтобы можно было свернуться до A

Опр. **Автоматом LR(1)-пунктов** расширенной грамматики $G = (\Sigma, \Gamma, P, S')$ называется λ-НКА $I^1_G = (\Sigma \cup \Gamma, I', \delta, i'_o, I')$, где :

- $i'_0 = [S' \to \cdot S, \dashv]$
- δ — множество переходов вида:
  - $[A \to \beta_1 \cdot x\beta_2, a] \xrightarrow{x} [A \to \beta_1 x \cdot \beta_2, a]$ — базисные
  - $[A \to \beta_1 \cdot B\beta_2, a] \xrightarrow{\lambda} [B \to \cdot\beta, b], \ \ b \in FIRST(\beta_2 a)$ — λ-переходы

> Прежде чем сделать переход, надо что-то свернуть. За тем, что выводится из B, должно следовать то, с чего начинается $\beta_2 a$

Детерминированный автомат теперь будет огромный, так как пункты с одинаковыми ядрами но разными символами должны быть разнесены по разным состояниям.

Построим автомат для грамматики, на которой не сработал SLR(1) автомат

$S' \to \cdot S, \dashv$

Начинаем замыкать:

$S \to \cdot ac, \dashv \; S \to \cdot bDc, \dashv \; S \to \cdot Da, \dashv$

$D \to \cdot a, a$ — взяли FIRST(a)

Фото 26.09 ~13:58

Надо быть осторожнее с состояниями, в которых возможны и перенос, и свёртка. Когда переносим, нам пофиг на символ, смотрим только на ядро.

A^1_G — LR(1)-ДКА

$[A \to \beta \cdot, a] \in I \Rightarrow ACTION(I, a) = № (A \Rightarrow \beta)$

| ACTION | GOTO |

	a	b	c	$\dashv$	S	D
S				✓		
a^1	(4)		c^1			
a^2			(4)			
a^3				(3)		
b	a^2					D^2
c^1				(1)		
c^2				(2)		
D^1	a^3					
D^2			c^2			
∇	a^1	b			S	D^1