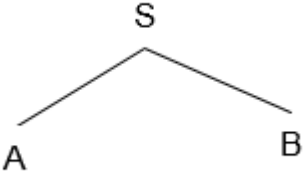


# Рекурсивный спуск

Recursive Descent

$S \rightarrow AB$      $A \rightarrow aA \mid a$      $B \rightarrow bB \mid b$

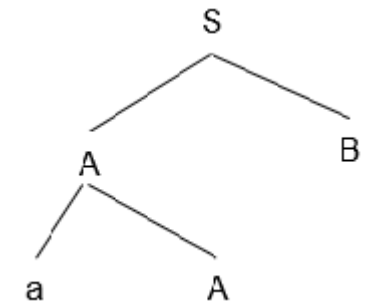
aaabb	$\epsilon$  AB	$S \rightarrow AB$	 <pre>graph TD; S --&gt; A; S --&gt; B;</pre>
-------	----------------------	--------------------	----------------------------------------------------------------------------------------------------------------------------------

$S \rightarrow AB$      $A \rightarrow aA \mid a$      $B \rightarrow bB \mid b$

$aaabb$

$AB$

$A \rightarrow aA$



$a$  $aabb$

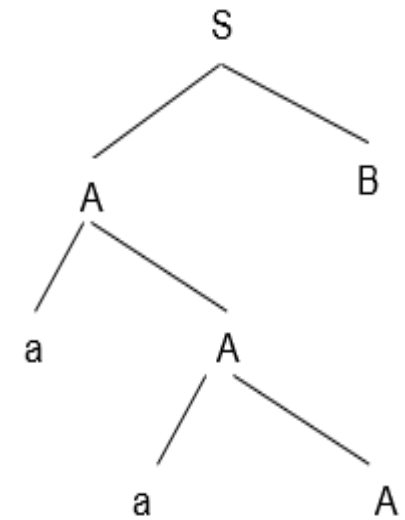
$aAB$

$S \rightarrow AB$      $A \rightarrow aA \mid a$      $B \rightarrow bB \mid b$

aaabb

aAB

$A \rightarrow aA$



aaAB

aaabb

$S \rightarrow AB$      $A \rightarrow aA \mid a$      $B \rightarrow bB \mid b$

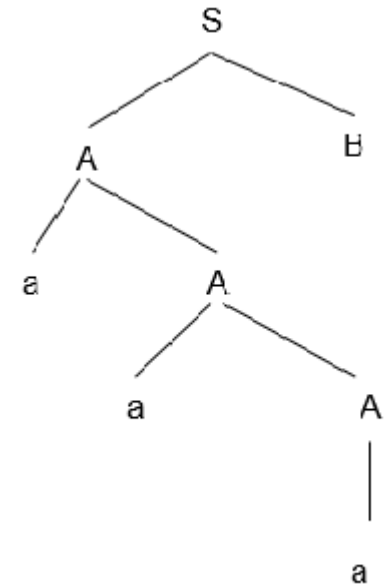
aaa**bb**

aaabb

aaAB

aaaB

$A \rightarrow a$



$S \rightarrow AB$      $A \rightarrow aA \mid a$      $B \rightarrow bB \mid b$

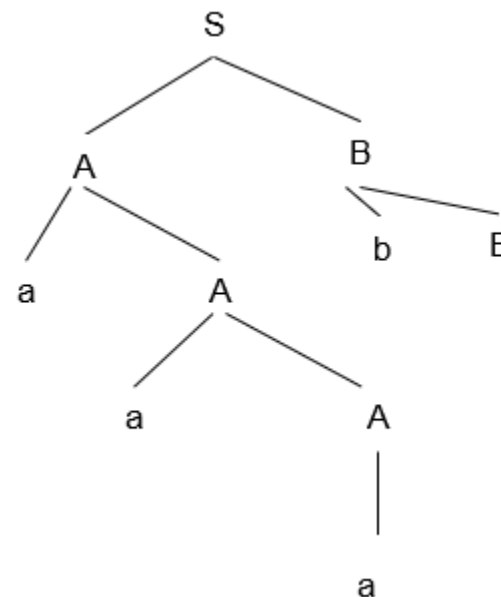
aaabb

aaabb

aaaB

aaabB

$B \rightarrow bB$



$S \rightarrow AB$      $A \rightarrow aA \mid a$      $B \rightarrow bB \mid b$

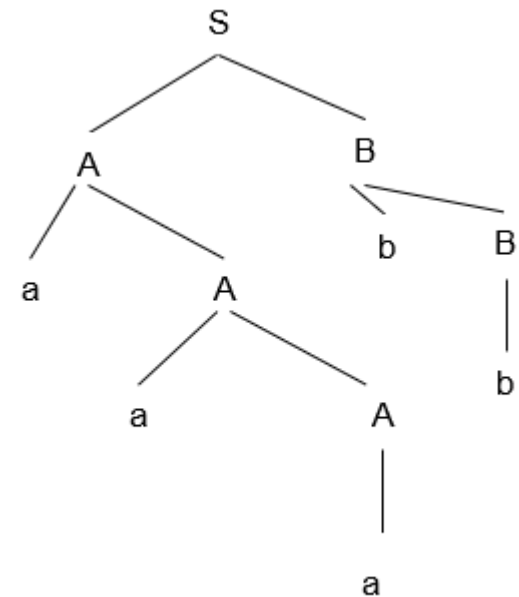
aaabb

aaabbb#

aaabB

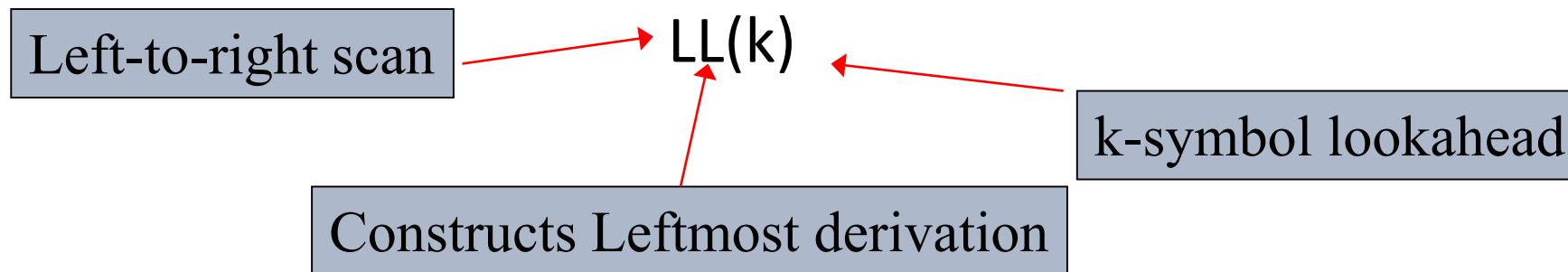
aaabb

$B \rightarrow b$



# LL(k) Грамматики

- класс грамматик: LL(k)



- Грамматика из примера - LL(2)
- На практике широко используются только LL(1)

$S \rightarrow AB$	$A \rightarrow aA \mid a$	$B \rightarrow bB \mid b$
--------------------	---------------------------	---------------------------



# Рекурсивный спуск (всё хорошо)

- Грамматика

$S \rightarrow ABd$

$A \rightarrow a \mid cA$

$B \rightarrow bA$

$A \rightarrow \alpha$  где  $\alpha \in (VT \cup VN)^*$

$A \rightarrow a_1\alpha_1 \mid a_2\alpha_2 \mid \dots \mid a_n\alpha_n,$

$a_i \in VT \ a_i \neq a_j \ i \neq j \ \alpha_i \in (VT \cup VN)^*$

- Принцип разбора

вывод начинается с  $S \rightarrow ABd$

СФ  $wB\alpha$ , где  $w \in T^*$   $B$  — самый левый нетерминал то  $B \rightarrow bA$

СФ  $wA\alpha$ , где  $w \in T^*$   $A$  — самый левый нетерминал то

$\alpha = a\alpha_1 \quad A \rightarrow a$

$\alpha = c\alpha_1 \quad A \rightarrow cA$

LL(1)

# Рекурсивный спуск (не всё хорошо)

- Грамматика

$$S \rightarrow aA \mid B \mid d$$

$$A \rightarrow d \mid aA$$

$$B \rightarrow aA \mid a$$

$$A \rightarrow a$$

$$B \Rightarrow^* a$$

$$S \rightarrow aA \text{ или } S \rightarrow B$$

- Грамматика

$$S \rightarrow A \mid B$$

$$A \rightarrow aA \mid c$$

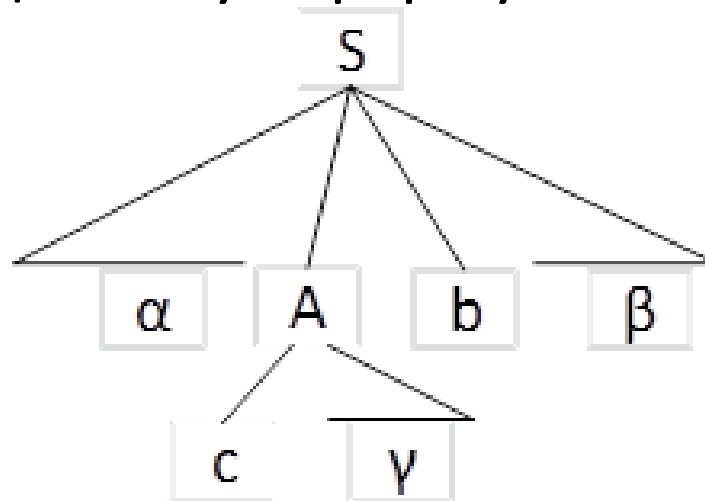
$$B \rightarrow aB \mid d$$

$a \dots \dots \dots ad$  <- Понятно!

Нужно раскрывать  $S \rightarrow B$

# Множество FIRST

Рассмотрим сентенциальную форму  $S \Rightarrow^* \alpha A b \beta$  ( $A \Rightarrow^+ c \gamma$ )



Множество  $\text{FIRST}(A)$  – это множество терминальных символов, с которых начинаются цепочки, выводимые из  $A$ .

Расширим для цепочек.

$$\text{FIRST}(\alpha) = \{ a \in VT \mid \alpha \Rightarrow^* a\alpha_1, \alpha_1 \in (VTUVN)^* \}$$

# Множество FIRST

$S \rightarrow A\#$     $A \rightarrow TB$     $B \rightarrow Z \mid \varepsilon$     $Z \rightarrow +TY$     $Y \rightarrow Z \mid \varepsilon$     $T \rightarrow b \mid (A)$

Номер шага	FIRST(S)	FIRST(A)	FIRST(B)	FIRST(Z)	FIRST(Y)	FIRST(T)
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\{\varepsilon\}$	$\{+\}$	$\{\varepsilon\}$	$\{b, (\}$
2	$\emptyset$	$\{b, (\}$	$\{\varepsilon, +\}$	$\{+\}$	$\{\varepsilon, +\}$	$\{b, (\}$
3	$\{b, (\}$	$\{b, (\}$	$\{\varepsilon, +\}$	$\{+\}$	$\{\varepsilon, +\}$	$\{b, (\}$
4	$\{b, (\}$	$\{b, (\}$	$\{\varepsilon, +\}$	$\{+\}$	$\{\varepsilon, +\}$	$\{b, (\}$

# FIRST для строки

$$\text{FIRST}(\alpha) = \{ a \in VT \mid \alpha \Rightarrow^* a\alpha_1, \alpha_1 \in (VT \cup VN)^* \}$$

Расширим содержание функции isnullable следующим образом:

isnullable(t) = false for  $t \in VT$

isnullable( $\epsilon$ ) = true

isnullable( $b_1b_2 \dots b_k$ ) = nullable( $b_1$ )  $\wedge$  nullable( $b_2$ )  $\wedge \dots \wedge$  nullable( $b_k$ );

тогда

$\text{FIRST}(A\alpha) = \text{FIRST}(A)$  если isnullable(A) == false

$\text{FIRST}(A\alpha) = \text{FIRST}(A) \cup \text{FIRST}(\alpha)$  если isnullable(A) == true.

# Множество FIRST и рекурсивный спуск

Если в грамматике существуют альтернативные правые части для некоторого нетерминала  $A \rightarrow \alpha \mid \beta$  такие, что  $\text{FIRST}(\alpha) \cap \text{FIRST}(\beta) \neq \emptyset$ , то метод рекурсивного спуска **неприменим** для этой грамматики.

Что делать? Преобразовывать грамматику.

Не существует алгоритма, определяющего для произвольной КС-грамматики, существует ли для нее эквивалентная грамматика, к которой применим метод рекурсивного спуска

(т. е. это алгоритмически неразрешимая проблема)

# Левая рекурсия.

Если в грамматике есть правило вида  $A \rightarrow Aa$ , то метод рекурсивного спуска неприменим.

- Устранение прямой левой рекурсии.

$A \rightarrow A\alpha_1 \mid \dots \mid A\alpha_n \mid \beta_1 \mid \dots \mid \beta_m$ , где

- $\alpha_i \in (VT \cup VN)^+$  для  $i = 1, 2, \dots, n$ ;
- $\beta_i \in (VT \cup VN)^*$

то левую рекурсию можно заменить правой:

- $A \rightarrow \beta_1 A' \mid \dots \mid \beta_m A'$
- $A' \rightarrow \alpha_1 A' \mid \dots \mid \alpha_n A' \mid \epsilon$  < ----- !!!!!

# Устранение прямой левой рекурсии.

- Исходное правило

$$A \rightarrow Aa \mid Ab \mid Ac \mid d \mid e$$

- Преобразованное правило.

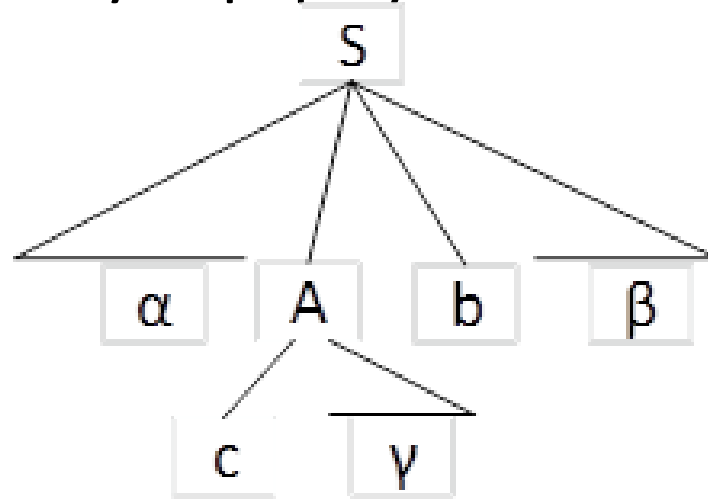
$$A \rightarrow \begin{array}{l} dX \\ | eX \end{array}$$
$$X \rightarrow \begin{array}{l} aX \\ | bX \\ | cX \\ | \epsilon \end{array}$$

Как реализовать процедуру для X?



# Множество FOLLOW

Рассмотрим sentential form  $S \Rightarrow^* \alpha A b \beta$  ( $A \Rightarrow^+ c \gamma$ )



Множество  $\text{FOLLOW}(A)$  для нетерминала  $A$  определяется как множество терминальных символов  $b$ , которые в sentential forms располагаются непосредственно справа от  $A$

# Множество FOLLOW

---

Алгоритм 4.13 Построение множества FOLLOW

Вход:  $G = (VT, VN, P, S)$

Выход: множества FOLLOW для нетерминалов грамматики  $G$

```
foreach A in VN do FOLLOW(A) = {}
FOLLOW(S) = {$}
flag = 1
while(flag) {
    flag = 0
    foreach A in VN {
        foreach  $B \rightarrow \alpha A \beta$  in P{
            FOLLOW(A) = FOLLOW(A)  $\cup$  FIRST( $\beta$ );
            if isnullable( $\beta$ )
                FOLLOW(A) = FOLLOW(A)  $\cup$  FOLLOW(B);
            if FOLLOW(A) изменилось flag = 1
        }
    }
}
```

## Множество FOLLOW

$S \rightarrow A$     $A \rightarrow TB$     $B \rightarrow Z \mid \varepsilon$     $Z \rightarrow +TY$     $Y \rightarrow Z \mid \varepsilon$     $T \rightarrow b \mid (A)$

$\text{FOLLOW}(A) = \{a \in VT \mid S \Rightarrow^* \alpha A \beta, \beta \Rightarrow^* a \gamma, A \in VN, \alpha, \beta, \gamma \in (VT \cup VN)^*\}$

	FLLW(S)	FLLW(A)	FLLW(B)	FLLW(T)	FLLW(Y)	FLLW(Z)
0	\$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	\$	{ \$, ) }	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	\$	{ \$, ) }	{ \$, ) }	$\emptyset$	$\emptyset$	$\emptyset$
3	\$	{ \$, ) }	{ \$, ) }	{ \$, + }	$\emptyset$	{ \$, ) }
4	\$	{ \$, ) }	{ \$, ) }	{ \$, +, ) }	$\emptyset$	{ \$, ) }
5	\$	{ \$, ) }	{ \$, ) }	{ \$, +, ) }	{ \$, ) }	{ \$, ) }
6	\$	{ \$, ) }	{ \$, ) }	{ \$, +, ) }	{ \$, ) }	{ \$, ) }

# Процедура для X

- Правило

$$\begin{array}{l} X \rightarrow a X \\ \quad | b X \\ \quad | c X \\ \quad | \epsilon \end{array}$$

```
void X()
{
    if (token == 'a' )
        {token = getchar(); X(); }
    else if (token == 'b')
        {token = getchar(); X(); }
    else if (token == 'c')
        {token = getchar(); X(); }
    else if (token ∈ FOLLOW(X))
        ;
    else Error();
}
```

# Множество FOLLOW (продолжение)

Если в грамматике есть правила  $A \rightarrow \alpha \mid \beta$ , такие что  $\beta \Rightarrow^* \varepsilon$ , и  $\text{FIRST}(\alpha) \cap \text{FOLLOW}(\beta) \neq \emptyset$ , то метод рекурсивного спуска **неприменим** к данной грамматике

А если  $A \rightarrow \alpha \mid \beta$ , такие что  $\alpha \Rightarrow^* \varepsilon$  и  $\beta \Rightarrow^* \varepsilon$  – тем более не допустим.

# Левая факторизация.

$$A \rightarrow a\alpha_1 \mid a\alpha_2 \mid \dots \mid a\alpha_n \mid \beta_1 \mid \dots \mid \beta_m$$

где  $a \in VT$ ;  $\alpha_i, \beta_j \in (VT \cup VN)^*$ ;

$$A \rightarrow aA' \mid \beta_1 \mid \dots \mid \beta_m$$

$$A' \rightarrow \alpha_1 \mid \alpha_2 \mid \dots \mid \alpha_n$$

- Правило

$$A \rightarrow aA \mid aB \mid aC \mid b \mid c$$

- После левой факторизации

$$A \rightarrow aX \mid b \mid c$$

$$X \rightarrow A \mid B \mid C$$

# Левая факторизация. Внимание!

- Правило

$$A \rightarrow aA \mid aB \mid aC \mid a \mid b$$

- После левой факторизации

$$A \rightarrow aX \mid b$$

$$X \rightarrow A \mid B \mid C \mid \epsilon$$

Появилось  $\epsilon$  – правило!

# ПРИМЕР

- Исходная грамматика

$S \rightarrow ABCd$

$A \rightarrow Aa \mid Ac \mid a$

$B \rightarrow b \mid bB$

$C \rightarrow c$

- После устранения левой рекурсии и левой факторизации

•  $S \rightarrow ABCd$

•  $A \rightarrow a X$

•  $B \rightarrow b Y$

•  $X \rightarrow a X \mid cX \mid \epsilon$

•  $Y \rightarrow B \mid \epsilon$

•  $C \rightarrow c$



# ПРИМЕР

Реализация.

```
void S()
{
    A();
    B();
    C();
    if (token == 'd')
    {token = getchar();
     if (token != '#')
        Error(); }
}
```

```
void A()
{
    if (token == 'a')
        {token = getchar();
         X(); }
    else Error();
}
B() и C() - аналогично
```

```
void X()
{
    if (token == 'a' )
    {token = getchar(); X(); }
    else if (token == 'c')
    {token = getchar(); X(); }
    else if (token == 'b'))
        ;
    else Error();
}
```

# ПРИМЕР

Как быть с

$Y \rightarrow B \mid \varepsilon$  ?

	Nullable	First	Follow
A		a	b
B		b	c
C		c	d
S		A	\$
X	1	a c $\varepsilon$	b
Y	1	b $\varepsilon$	c

Для  $Y \rightarrow B \mid \varepsilon$

$\text{FIRST}(B) = \{b\}$   $\text{FOLLOW}(B) = \{c\}$

$\text{FIRST}(B) \cap \text{FOLLOW}(B) = \emptyset$

void Y()

{

if (token == 'b' )

{token = getchar(); B(); }

else if (token == 'c')

;

else Error();

}

## Восходящий распознаватель

- Реализуется восходящий **распознаватель** с использованием стека (МП-автомат):
- На каждом шаге производится
  - **сдвиг**: текущий символ помещается в стек, читается следующий символ; или
  - **свертка**: если верхние  $m$  символов стека соответствуют правой части правила, они заменяются на левую часть правила (единственный нетерминал).
- **Успех (цепочка принята)** если на вершине стека остался единственный символ: начальный символ грамматики

## Распознаватель типа Сдвиг-Свертка (2)

$E_x \rightarrow \text{Nat} \mid (E_x) \mid E_x + E_x \mid E_x * E_x$
-----------------------------------------------------------------------

<u>СТЕК</u>	<u>СОСТОЯНИЕ ВХОДА</u>	<u>ДЕЙСТВИЕ</u>
	Nat + Nat * Nat	←shift
Nat	+ Nat * Nat	↓reduce
E <sub>x</sub>	+ Nat * Nat	←shift
E <sub>x</sub> +	Nat * Nat	←shift
E <sub>x</sub> + Nat	* Nat	↓reduce
E <sub>x</sub> + E <sub>x</sub>	* Nat	←shift
E <sub>x</sub> + E <sub>x</sub> *	Nat	←shift
E <sub>x</sub> + E <sub>x</sub> * Nat		↓reduce
E <sub>x</sub> + E <sub>x</sub> * E <sub>x</sub>		↓reduce
E <sub>x</sub> + E <sub>x</sub>		↓reduce
E <sub>x</sub>		ACCEPT