

# 自顶向下的语法分析



WANG Hanfei

School of Computer  
Wuhan University

October 13, 2015

## 1 基本概念

- 引言
- 自顶向下的语法分析
- 文法的等价变换

## 2 预测分析法

- 基本概念
- 递归下降分析法
- LL(1) 分析法

## 3 LL(1) 文法

- 首符号集和后随符号集
- LL(1) 分析表的构造
- LL(1) 语言

## 4 LL(1) 分析法出错处理

- 相关概念
- 同步符号
- 非同步符号的处理

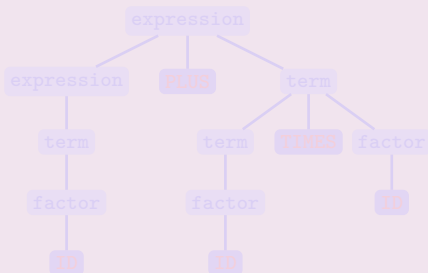
# XL 语言

1 + a \* 3

词法  
分析

ID  
PLUS  
ID  
TIMES  
ID

语法  
分析



## XL 语言

词法  
→  
分析

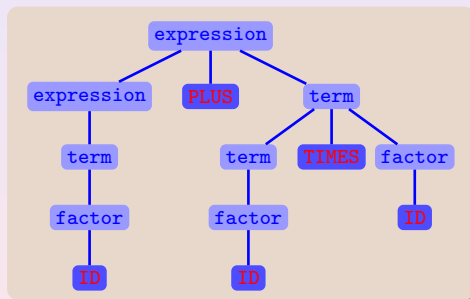
ID  
PLUS  
ID  
TIMES  
ID

## XL 语言

1 + a \* 3

词法  
分析

ID
PLUS
ID
TIMES
ID

语法  
分析

## 语法分析 (Parser)

- 形式语言的第二次抽象，对单词再次重组。
  - Ex:  $\Sigma = \{\text{ID, PLUS, TIMES, LP, RP, SEMI}\}$ ;
  - $XL \subseteq \Sigma^*$ .
- 描述问题：
  - 正则表达式：表达能力太弱，嵌套的括号对不能描述.
  - 数学递归定义：不利于机器对文法的识别.
- 识别问题:  $\exists P \Sigma^* \rightarrow \text{Boolean}, s \mapsto \text{true or false}.$
- 语法树的建立.

## 自顶向下的语法分析 (Top-down Parsing).

- 从左到右对输入的终结符进行扫描.
- 对输入的语句寻找到一个最左推导.
- 对输入的语句自顶向下建立语法树模拟最左推导.
- 无回溯(Backtracking): 分析出错一定是输入的语句出错, 而不是分析过程的推导出错.
- 预测分析法(predictive parser): 根据当前输入和当前需要展开的非终结符选择唯一的产生式.

### Predictive Parser

- 递归下降分析法(Recursive descent parsing), 如: XL 语言.
- LL(1) 分析法 (表驱动).
- 共同特点: match() --- advance().
- 对文法有特定的要求.

## 选择产生式的二难

## Example

$T = \{\text{ID}, \text{PLUS}, \text{TIMES}, \text{LP}, \text{RP}\}$

$N = \{\text{exp}, \text{term}, \text{fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp} \text{ PLUS term}$

2/     |  $\text{term}$

3/  $\text{term} \rightarrow \text{term} \text{ TIMES fac}$

4/     |  $\text{fac}$

5/  $\text{fac} \rightarrow \text{ID}$

6/     |  $\text{LP exp RP}$

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
1	$\Rightarrow$	
lm	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
2	$\Rightarrow$	
lm	<u>term</u> PLUS term	ID PLUS ID TIMES ID
3	$\Rightarrow$	
lm	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
4	$\Rightarrow$	
lm	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
5	$\Rightarrow$	
lm	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
6	$\Rightarrow$	
lm	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
7	$\Rightarrow$	
lm	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
8	$\Rightarrow$	
lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

## 推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式1/和2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式3/和4/都可以推出以 ID 为首符号的句型。



## 选择产生式的二难

## Example

$T = \{\text{ID}, \text{PLUS}, \text{TIMES}, \text{LP}, \text{RP}\}$

$N = \{\text{exp}, \text{term}, \text{fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp} \text{ PLUS term}$

2/     | term

3/  $\text{term} \rightarrow \text{term} \text{ TIMES fac}$

4/     | fac

5/  $\text{fac} \rightarrow \text{ID}$

6/     | LP exp RP

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
1 $\Rightarrow$ lm	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
2 $\Rightarrow$ lm	<u>term</u> PLUS term	ID PLUS ID TIMES ID
3 $\Rightarrow$ lm	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
4 $\Rightarrow$ lm	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
5 $\Rightarrow$ lm	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
6 $\Rightarrow$ lm	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
7 $\Rightarrow$ lm	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
8 $\Rightarrow$ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

## 推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式 1/和 2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式 3/和 4/都可以推出以 ID 为首符号的句型。

## 选择产生式的二难

## Example

$T = \{\text{ID}, \text{PLUS}, \text{TIMES}, \text{LP}, \text{RP}\}$

$N = \{\text{exp}, \text{term}, \text{fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp} \text{ PLUS term}$

2/     |  $\text{term}$

3/  $\text{term} \rightarrow \text{term} \text{ TIMES fac}$

4/     |  $\text{fac}$

5/  $\text{fac} \rightarrow \text{ID}$

6/     |  $\text{LP exp RP}$

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
1 ⇒ lm	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
2 ⇒ lm	<u>term</u> PLUS term	ID PLUS ID TIMES ID
3 ⇒ lm	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
4 ⇒ lm	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
5 ⇒ lm	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
6 ⇒ lm	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
7 ⇒ lm	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
8 ⇒ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

## 推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式1/和2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式3/和4/都可以推出以 ID 为首符号的句型。

## 选择产生式的二难

## Example

$T = \{\text{ID}, \text{PLUS}, \text{TIMES}, \text{LP}, \text{RP}\}$

$N = \{\text{exp}, \text{term}, \text{fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp} \text{ PLUS term}$

2/     | term

3/  $\text{term} \rightarrow \text{term} \text{ TIMES fac}$

4/     | fac

5/  $\text{fac} \rightarrow \text{ID}$

6/     | LP exp RP

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
1 $\Rightarrow$ lm	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
2 $\Rightarrow$ lm	<u>term</u> PLUS term	ID PLUS ID TIMES ID
3 $\Rightarrow$ lm	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
4 $\Rightarrow$ lm	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
5 $\Rightarrow$ lm	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
6 $\Rightarrow$ lm	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
7 $\Rightarrow$ lm	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
8 $\Rightarrow$ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

## 推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式1/和2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式3/和4/都可以推出以 ID 为首符号的句型。

## 选择产生式的二难

## Example

$T = \{\text{ID}, \text{PLUS}, \text{TIMES}, \text{LP}, \text{RP}\}$

$N = \{\text{exp}, \text{term}, \text{fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp} \text{ PLUS term}$

2/     | term

3/  $\text{term} \rightarrow \text{term} \text{ TIMES fac}$

4/     | fac

5/  $\text{fac} \rightarrow \text{ID}$

6/     | LP exp RP

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
$\xRightarrow{1}$ lm	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{2}$ lm	<u>term</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{3}$ lm	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{4}$ lm	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
$\xRightarrow{5}$ lm	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
$\xRightarrow{6}$ lm	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
$\xRightarrow{7}$ lm	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
$\xRightarrow{8}$ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

## 推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式1/和2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式3/和4/都可以推出以 ID 为首符号的句型。

## 选择产生式的二难

## Example

$T = \{\text{ID, PLUS, TIMES, LP, RP}\}$

$N = \{\text{exp, term, fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp PLUS term}$

2/     |  $\text{term}$

3/  $\text{term} \rightarrow \text{term TIMES fac}$

4/     |  $\text{fac}$

5/  $\text{fac} \rightarrow \text{ID}$

6/     |  $\text{LP exp RP}$

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
1 $\Rightarrow$ lm	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
2 $\Rightarrow$ lm	<u>term</u> PLUS term	ID PLUS ID TIMES ID
3 $\Rightarrow$ lm	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
4 $\Rightarrow$ lm	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
5 $\Rightarrow$ lm	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
6 $\Rightarrow$ lm	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
7 $\Rightarrow$ lm	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
8 $\Rightarrow$ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式1/和2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式3/和4/都可以推出以 ID 为首符号的句型。

## 选择产生式的二难

## Example

$T = \{\text{ID}, \text{PLUS}, \text{TIMES}, \text{LP}, \text{RP}\}$

$N = \{\text{exp}, \text{term}, \text{fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp} \text{ PLUS term}$

2/     |  $\text{term}$

3/  $\text{term} \rightarrow \text{term} \text{ TIMES fac}$

4/     |  $\text{fac}$

5/  $\text{fac} \rightarrow \text{ID}$

6/     |  $\text{LP exp RP}$

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
1 $\Rightarrow$ lm	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
2 $\Rightarrow$ lm	<u>term</u> PLUS term	ID PLUS ID TIMES ID
3 $\Rightarrow$ lm	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
4 $\Rightarrow$ lm	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
5 $\Rightarrow$ lm	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
6 $\Rightarrow$ lm	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
7 $\Rightarrow$ lm	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
8 $\Rightarrow$ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式1/和2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式3/和4/都可以推出以 ID 为首符号的句型。

## 选择产生式的二难

## Example

$T = \{\text{ID}, \text{PLUS}, \text{TIMES}, \text{LP}, \text{RP}\}$

$N = \{\text{exp}, \text{term}, \text{fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp} \text{ PLUS term}$

2/     |  $\text{term}$

3/  $\text{term} \rightarrow \text{term} \text{ TIMES fac}$

4/     |  $\text{fac}$

5/  $\text{fac} \rightarrow \text{ID}$

6/     |  $\text{LP exp RP}$

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
$\xRightarrow{1}_{lm}$	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{2}_{lm}$	<u>term</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{3}_{lm}$	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{4}_{lm}$	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
$\xRightarrow{5}_{lm}$	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
$\xRightarrow{6}_{lm}$	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
$\xRightarrow{7}_{lm}$	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
$\xRightarrow{8}_{lm}$	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式1/和2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式3/和4/都可以推出以 ID 为首符号的句型。

## 选择产生式的二难

## Example

$T = \{\text{ID}, \text{PLUS}, \text{TIMES}, \text{LP}, \text{RP}\}$

$N = \{\text{exp}, \text{term}, \text{fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp} \text{ PLUS term}$

2/     |  $\text{term}$

3/  $\text{term} \rightarrow \text{term} \text{ TIMES fac}$

4/     |  $\text{fac}$

5/  $\text{fac} \rightarrow \text{ID}$

6/     |  $\text{LP exp RP}$

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
$\xRightarrow{1}$ lm	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{2}$ lm	<u>term</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{3}$ lm	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{4}$ lm	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
$\xRightarrow{5}$ lm	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
$\xRightarrow{6}$ lm	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
$\xRightarrow{7}$ lm	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
$\xRightarrow{8}$ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

## 推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式1/和2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式3/和4/都可以推出以 ID 为首符号的句型。



## 选择产生式的二难

## Example

$T = \{\text{ID}, \text{PLUS}, \text{TIMES}, \text{LP}, \text{RP}\}$

$N = \{\text{exp}, \text{term}, \text{fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp} \text{ PLUS term}$

2/     | term

3/  $\text{term} \rightarrow \text{term} \text{ TIMES fac}$

4/     | fac

5/  $\text{fac} \rightarrow \text{ID}$

6/     | LP exp RP

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
$\xRightarrow{1}_{lm}$	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{2}_{lm}$	<u>term</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{3}_{lm}$	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
$\xRightarrow{4}_{lm}$	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
$\xRightarrow{5}_{lm}$	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
$\xRightarrow{6}_{lm}$	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
$\xRightarrow{7}_{lm}$	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
$\xRightarrow{8}_{lm}$	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

## 推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式1/和2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式3/和4/都可以推出以 ID 为首符号的句型。

## 选择产生式的二难

## Example

$T = \{\text{ID, PLUS, TIMES, LP, RP}\}$

$N = \{\text{exp, term, fac}\}$

$S = \text{exp}$

P:

1/  $\text{exp} \rightarrow \text{exp PLUS term}$

2/     | term

3/  $\text{term} \rightarrow \text{term TIMES fac}$

4/     | fac

5/  $\text{fac} \rightarrow \text{ID}$

6/     | LP exp RP

## 最左推导

	<u>exp</u>	ID PLUS ID TIMES ID
1 $\Rightarrow$ lm	<u>exp</u> PLUS term	ID PLUS ID TIMES ID
2 $\Rightarrow$ lm	<u>term</u> PLUS term	ID PLUS ID TIMES ID
3 $\Rightarrow$ lm	<u>fac</u> PLUS term	ID PLUS ID TIMES ID
4 $\Rightarrow$ lm	ID PLUS <u>term</u>	ID PLUS ID TIMES ID
5 $\Rightarrow$ lm	ID PLUS <u>term</u> TIMES fac	ID PLUS ID TIMES ID
6 $\Rightarrow$ lm	ID PLUS <u>fac</u> TIMES fac	ID PLUS ID TIMES ID
7 $\Rightarrow$ lm	ID PLUS ID TIMES <u>fac</u>	ID PLUS ID TIMES ID
8 $\Rightarrow$ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID

## 推导过程中产生了选择产生式的不确定性

- 在推导的第1步和第4步，面对的当前输入都是 ID，产生式 1/和 2/都可以推出以 ID 为首符号的句型。
- 在推导的第2步和第6步，面对的当前输入都是 ID，产生式 3/和 4/都可以推出以 ID 为首符号的句型。

## 文法的等价变换

### 产生原因

- $\text{exp} \rightarrow \text{exp PLUS term} \mid \text{term}$  两个产生式产生的首符号都是  $\text{term}$ .
- $\text{term} \rightarrow \text{term TIMES fac} \mid \text{fac}$  两个产生式产生的首符号都是  $\text{fac}$ .
- $A \rightarrow A\alpha \mid \beta$ ,  $A$  的两个产生式推出的首符号都是  $\beta$ .
- $A \rightarrow aB \mid aC$ ,  $A$  的两个产生式推出的首符号都是  $a$ .

### 解决方法 --- 文法等价变换

使得非终结符  $\text{exp}$  要推出以  $\text{ID}$  为首句型只有一个产生式选择的可能.

## 消除左递归

## Example of XL

$$\text{exp} \rightarrow \text{exp} \text{ PLUS term} \mid \text{term}$$

对应的句型:

$$S = \{\text{term} \underbrace{(\text{PLUS term})^n}_{\text{exp}'} \mid n \in \mathbb{N}\}$$

句型的生成方式: 右边增长, 修改增长方式为: 左边增长:

$$\begin{aligned} \text{exp} &\rightarrow \text{term exp}' \\ \text{exp}' &\rightarrow \text{PLUS term exp}' \mid \varepsilon \end{aligned}$$

## 一般左递归文法

$$A \rightarrow A \alpha \mid \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$$

对应的句型:

$$\mathcal{A} = \{\beta_i \underbrace{\alpha^k}_{A'} \mid \beta_i = 1, \cdots, n, k \in \mathbb{N}\}$$

句型的生成方式: 右边增长, 修改增长方式为: 左边增长:

$$\begin{aligned} A &\rightarrow \beta_1 A' \mid \beta_2 A' \mid \cdots \mid \beta_n A' \\ A' &\rightarrow \alpha A' \mid \varepsilon \end{aligned}$$

## Example

### XL语言原文法

```
stmts -> exp SEMI EOI
        | exp SEMI stmts
```

```
exp -> exp PLUS term
      | term
```

```
term -> term TIMES fac
       | fac
```

```
fac -> ID
      | LP exp RP
```



### 消除左递归后的等价文法

```
stmts -> exp SEMI EOI
        | exp SEMI stmts
```

```
exp -> term exp'
```

```
exp' -> PLUS term exp'
       | epsilon /* 空串 */
```

```
term -> fac term'
```

```
term' -> TIMES fac term'
        | epsilon /* 空串 */
```

```
fac -> ID
      | LP exp RP
```

## 间接左递归

## 间接左递归

$$S \rightarrow Aa \mid b$$

$$A \rightarrow Ac \mid Sd \mid \varepsilon$$

产生间接左递归:

$$S \Rightarrow Aa \Rightarrow Sda$$

非终结符号分层, 开始符号层次最高, 产生终结符的层次最低:

$$S \rightarrow Aa \mid b$$

$$A \rightarrow Ac \mid \underbrace{Aa}_S d \mid \underbrace{b}_S d \mid \varepsilon$$

## 句型分析

$$S \rightarrow Aa \mid b$$

$$A \rightarrow A \underbrace{c}_{a_1} \mid A \underbrace{ad}_{a_2} \mid \underbrace{bd}_{\beta_1} \mid \underbrace{\varepsilon}_{\beta_2}$$

A 对应的句型:

$$\begin{aligned} \mathcal{A} &= \{(\beta_1 \mid \beta_2)(a_1 \mid a_2)^n \mid n \in \mathbf{N}\} \\ &= \{(\underbrace{bd \mid \varepsilon}_{A'})(c \mid ad)^n \mid n \in \mathbf{N}\} \end{aligned}$$

句型的生成方式: 右边增长, 修改增长方式为: 左边增长:

$$A \rightarrow A' \mid bdA'$$

$$A' \rightarrow cA' \mid adA' \mid \varepsilon$$

## 消除左公因子

## Example of if-then-else

$$\text{stmt} \rightarrow \underbrace{\text{if exp then stmt}}_{\text{commun factor}} \text{ else stmt}$$

$$| \underbrace{\text{if exp then stmt}}_{\text{commun factor}}$$

等价文法:

$$\text{stmt} \rightarrow \underbrace{\text{if exp then stmt}}_{\text{commun factor}} \text{ stmt'}$$

$$\text{stmt'} \rightarrow \text{else stmt} \mid \epsilon$$

## 一般情况

$$A \rightarrow \alpha \gamma \mid \alpha \delta \mid \beta$$

等价文法:

$$A \rightarrow \alpha A' \mid \beta$$

$$A' \rightarrow \gamma \mid \delta$$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	exp	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
lm		
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
lm		
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
lm		
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
lm		
⇒	ID PLUS term exp'	ID PLUS ID TIMES ID \$
lm		
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
lm		
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
lm		
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES fac term' exp'	ID PLUS ID TIMES ID \$
lm		
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
lm		
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID exp'	ID PLUS ID TIMES ID \$
lm		
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$
lm		



# Example of XL

## 推导过程

### XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

exp	ID PLUS ID TIMES ID \$
⇒ <u>term</u> exp'	ID PLUS ID TIMES ID \$
lm	
⇒ <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
lm	
⇒ ID term' exp'	ID PLUS ID TIMES ID \$
lm	
= ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ ID <u>exp'</u>	ID PLUS ID TIMES ID \$
lm	
⇒ ID PLUS term exp'	ID PLUS ID TIMES ID \$
lm	
= ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒ ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
lm	
⇒ ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
lm	
= ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ ID PLUS ID TIMES fac term' exp'	ID PLUS ID TIMES ID \$
lm	
= ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒ ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
lm	
= ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ ID PLUS ID TIMES ID exp'	ID PLUS ID TIMES ID \$
lm	
⇒ ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$
lm	

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒ lm	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS term exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID TIMES fac term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID TIMES ID exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS term exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES fac term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒ lm	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒ lm	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## 推导过程

### XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

Match-Advance

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS term exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES fac term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS term exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES fac term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$



# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

Match-Advance

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

Match-Advance

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>TIMES</u> fac term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

Match-Advance



# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## 推导过程

### XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

Match-Advance

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## 推导过程

### XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# Example of XL

## XL 语言文法

exp → term exp'

exp' → PLUS term exp'  
| epsilon  
/\* 空串 \*/

term → fac term'

term' → TIMES fac term'  
| epsilon  
/\* 空串 \*/

fac → ID  
| LP exp RP

## 推导过程

	<u>exp</u>	ID PLUS ID TIMES ID \$
⇒	<u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	<u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID term' exp'	ID PLUS ID TIMES ID \$
=	ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
=	ID PLUS <u>term</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES <u>fac</u> term' exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID term' exp'	ID PLUS ID TIMES ID \$
=	ID PLUS ID TIMES ID <u>term'</u> exp'	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID <u>exp'</u>	ID PLUS ID TIMES ID \$
⇒	ID PLUS ID TIMES ID	ID PLUS ID TIMES ID \$

# 预测分析法

## Predictive Parser

- 从左到右对输入的终结符扫描。
  - 句型的开始状态是文法开始符号。
  - 扫描的第一个终结符。
- 根据当前的终结符  $a$  和当前句型的符号  $X$ , 执行如下操作:
  - if ( $X \in T$  and  $X == a$ ) then  
读下个终结符; 句型取下一个符号.  
else 分析出错.
  - if ( $X \notin T$ ), 设  $A = X$ :  
选择唯一的一个产生式  $A \rightarrow \gamma$ , 该产生式满足:  
 $A \Rightarrow \gamma \xRightarrow{*} a\delta$ , 将句型中的  $X$  替换为  $\gamma$ , 并且置当前句型的符号为  $\gamma$  的第一个符号.
- 重复上步直到状态句型中没有符号要处理并且当前输入是文件结束标记.

# 预测分析法

## 对文法的要求

产生式  $A \rightarrow a_1 \mid \cdots \mid a_n$  只有唯一的一个能展开为  $a$  的句型.

## Predictive Parser

- 递归下降分析法 (Recursive descent parsing), XL 语言.
- LL(1) 分析法 (表驱动) (Left-to-right scan and Leftmost derivation and lookahead 1 symbol).



## 递归下降分析法 --- XL 语言

## 递归函数

```

/* exp -> term exp' */
void exp(void)
{
    term();
    expr_prime();
}

/*
 * exp' -> PLUS term exp'
 *      | epsilon
 */
void expr_prime(void)
{
    if ( match(PLUS) ) {
        advance();
        term();
        expr_prime();
    }
}

```



## while 循环

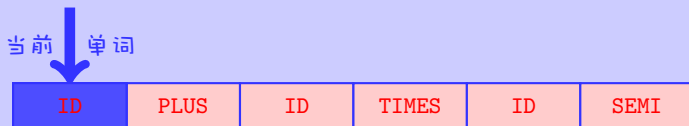
```

/* exp -> term
 *      (PLUS term exp')*
 */

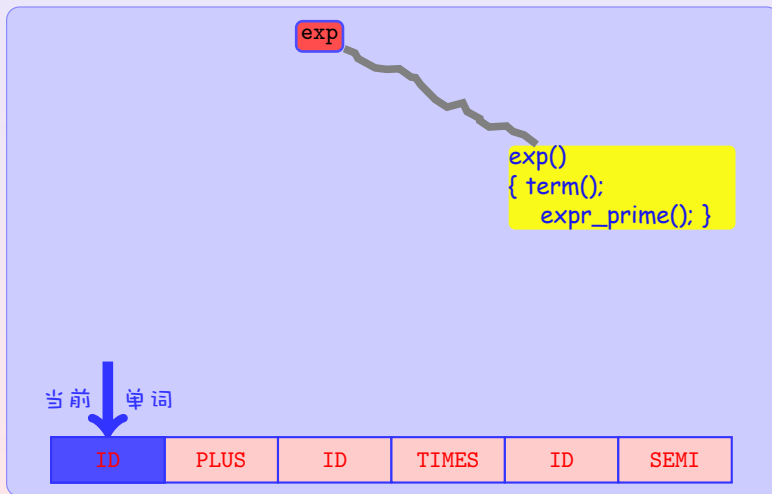
exp()
{
    term();
    while (match(PLUS)) {
        advance();
        term();
    }
}

```

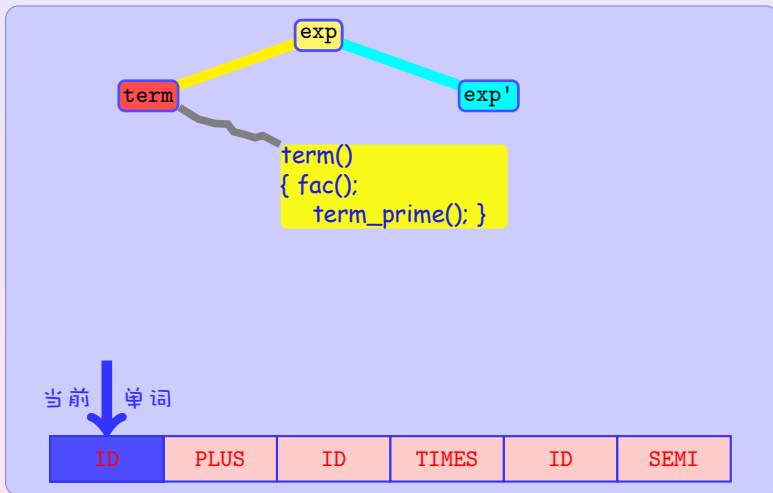
# 递归下降分析法演示



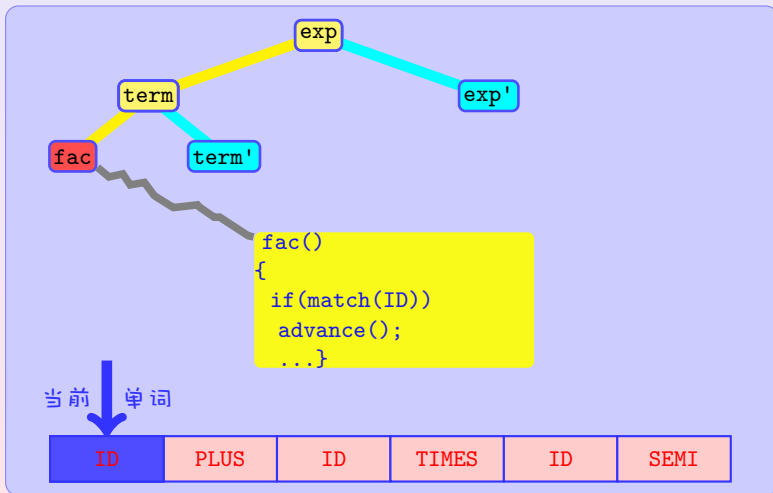
## 递归下降分析法演示



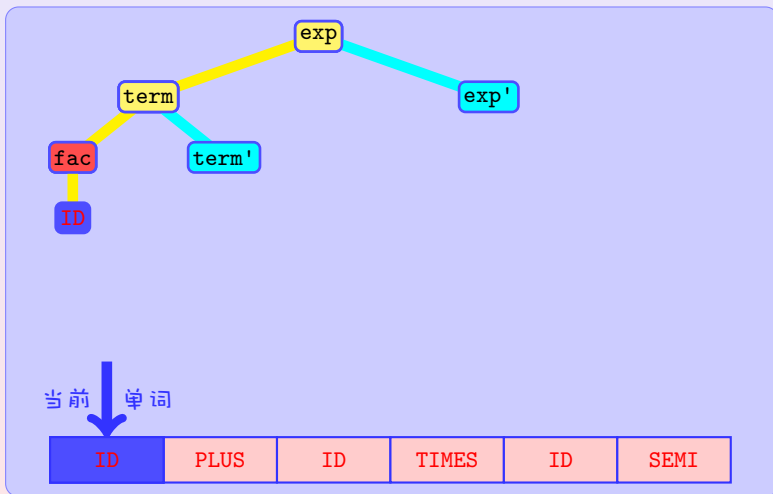
## 递归下降分析法演示



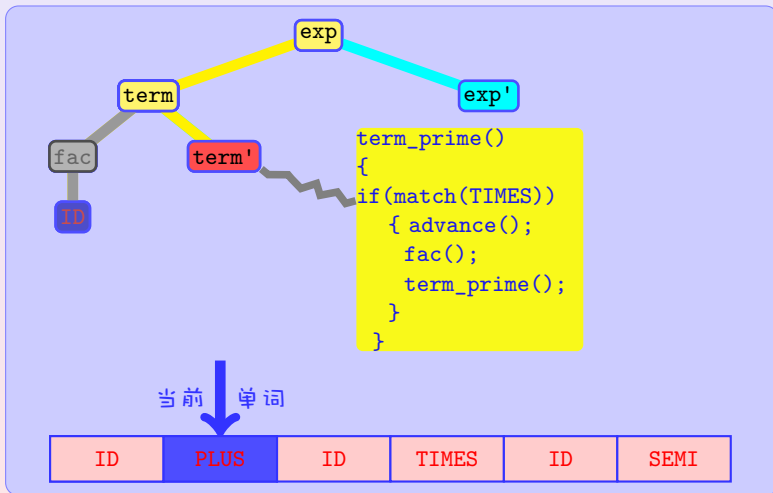
## 递归下降分析法演示



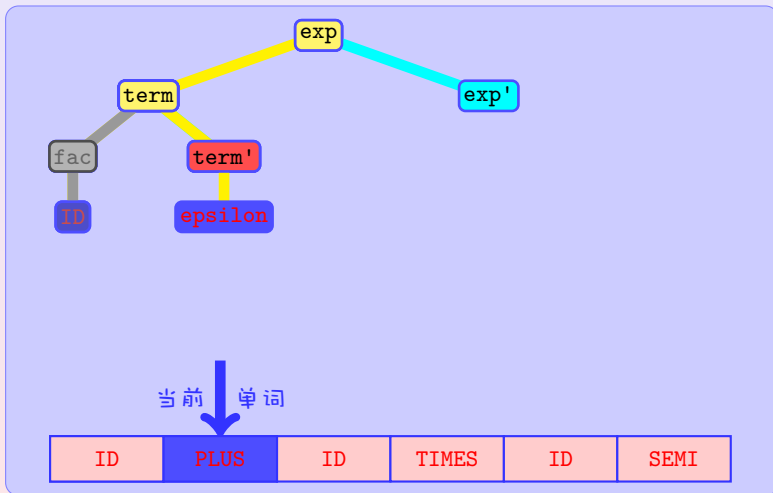
## 递归下降分析法演示



## 递归下降分析法演示

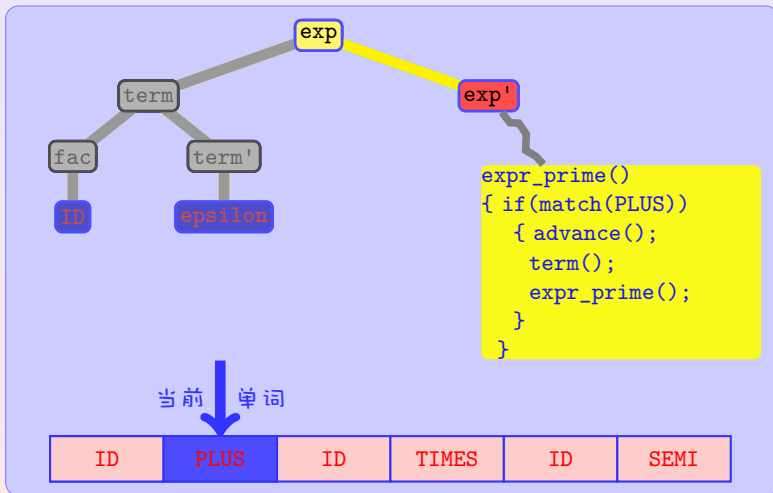


## 递归下降分析法演示

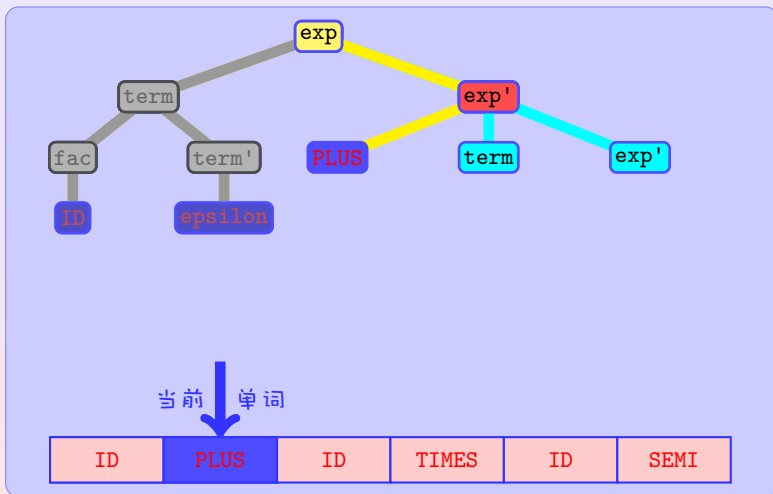




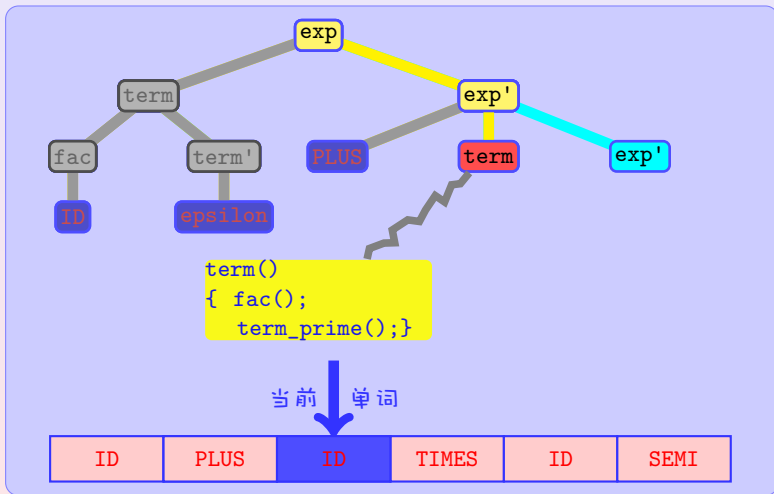
## 递归下降分析法演示



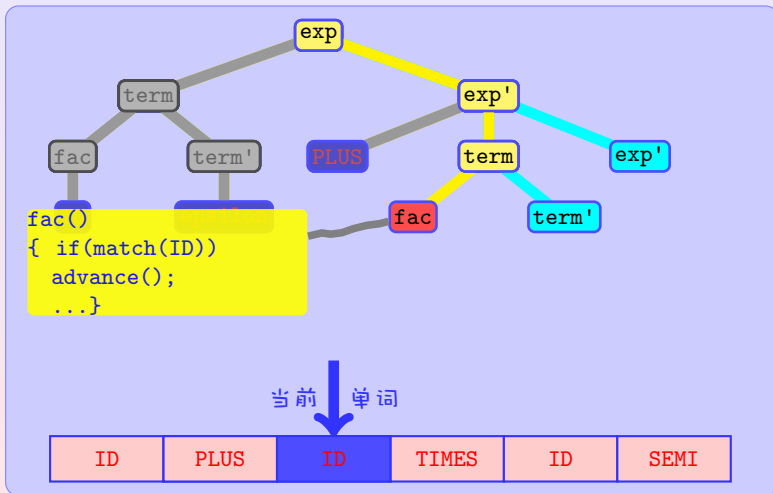
## 递归下降分析法演示



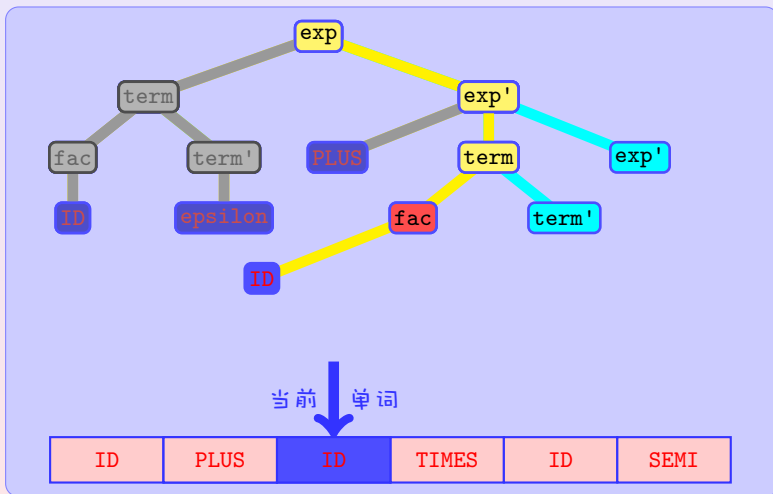
## 递归下降分析法演示



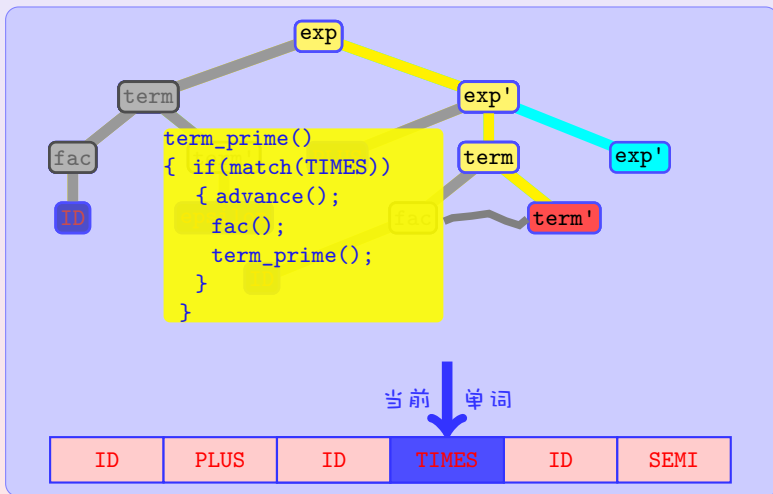
## 递归下降分析法演示



## 递归下降分析法演示

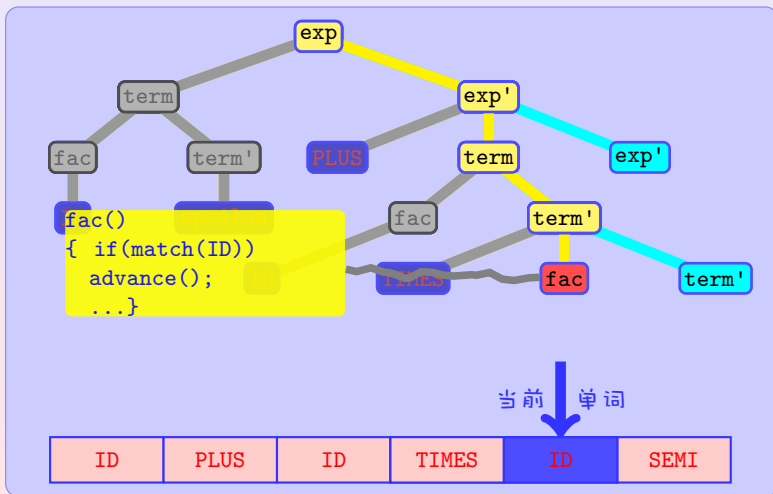


## 递归下降分析法演示



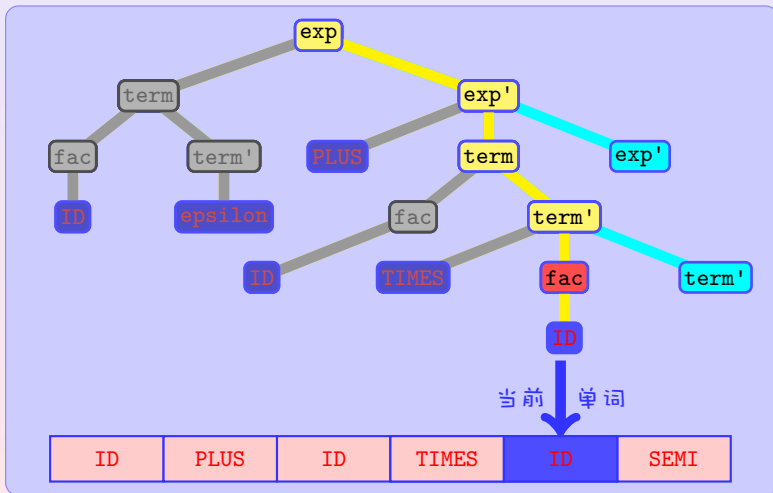


## 递归下降分析法演示

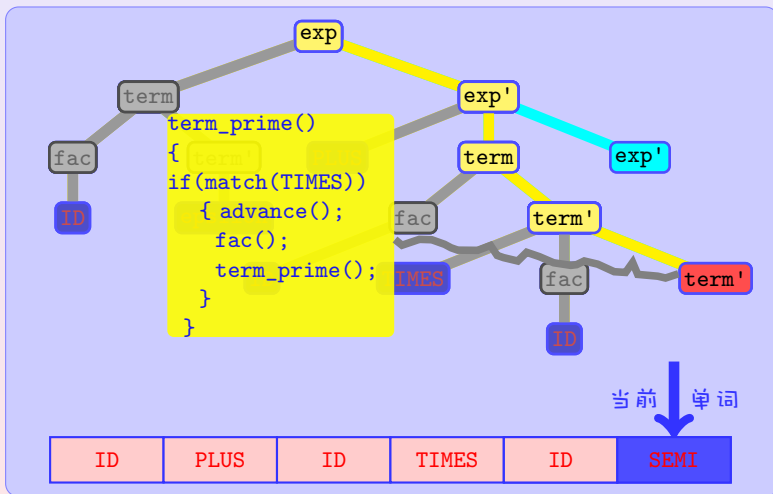




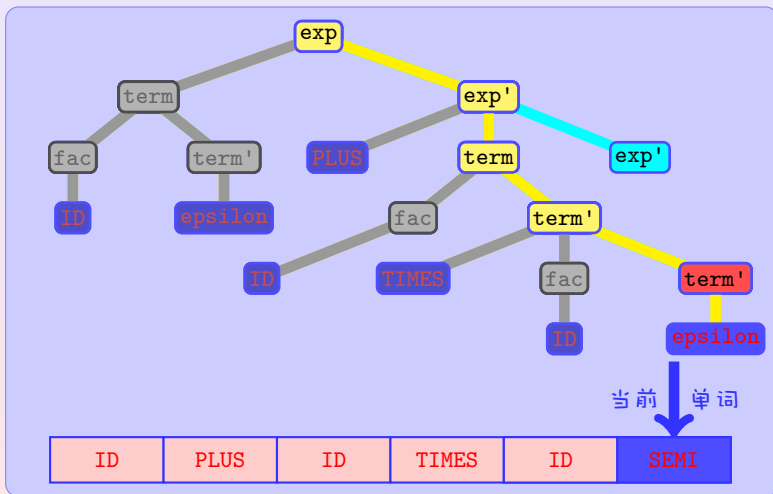
## 递归下降分析法演示



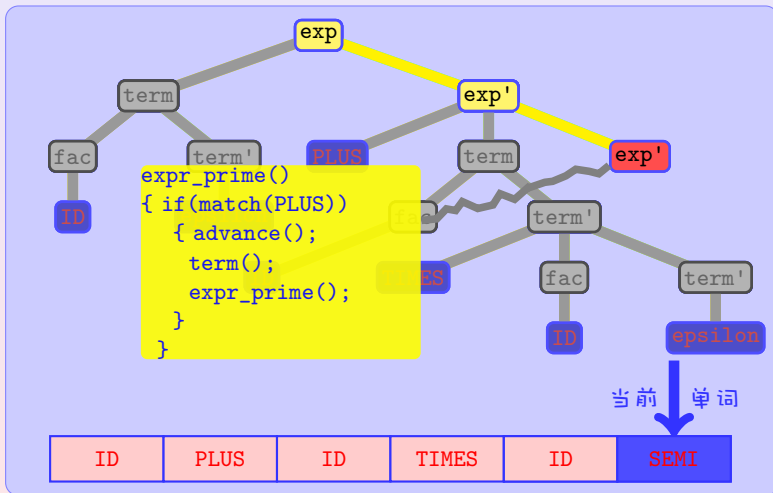
## 递归下降分析法演示



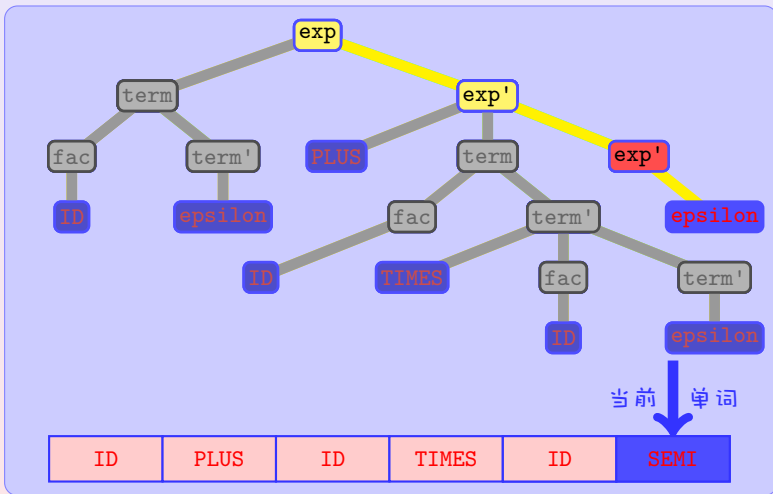
## 递归下降分析法演示



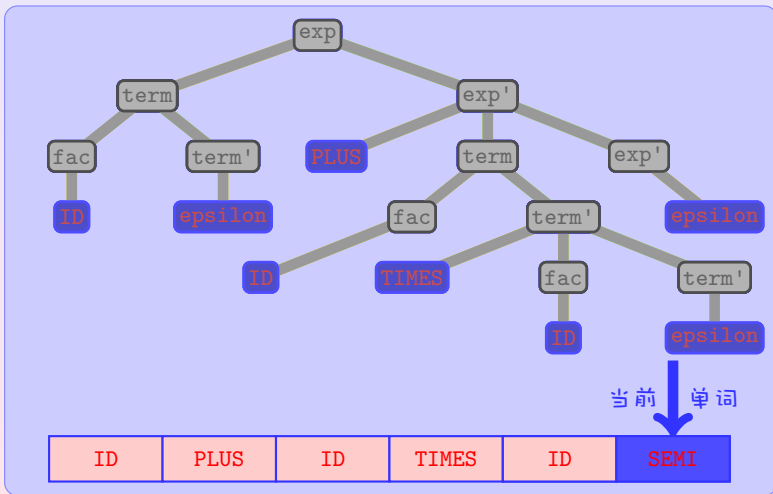
## 递归下降分析法演示



## 递归下降分析法演示



## 递归下降分析法演示



## 注释

### 原理

- 每个非终结符对应于一个函数，
- 每个终结符对应“match-advance”；
- 每个函数翻译为其对应的产生式右边符号串对应的调用序列；
- 多个可选的产生式，用分支语句（“if”；“switch”）；
- 函数调用过程和非终结符展开过程一致，函数的递归调用和语法树的前序遍历一致；

### 特点

- 程序简单；
- 递归调用增加内存开支。
- 适用较小规模的文法。

## 问题

$$\bullet \left\{ \begin{array}{l} S \rightarrow A \mid B \\ A \rightarrow a \\ B \rightarrow b \end{array} \right. \iff$$

S()

{

if (match(a)) { A(); return; }

if (match(b)) { B(); return; }

}

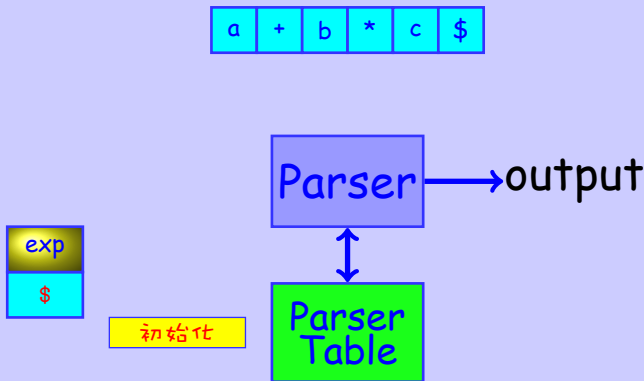
$$\bullet \left\{ \begin{array}{l} S \rightarrow AB \\ A \rightarrow aA \mid \varepsilon \\ B \rightarrow ac \mid b \end{array} \right. \Rightarrow \begin{array}{l} AB \Rightarrow aAB \\ \quad \quad \quad \text{lm} \\ AB \Rightarrow B \Rightarrow ac \\ \quad \quad \quad \text{lm} \quad \quad \text{lm} \end{array}$$

文法没有左递归和左公因子, 但是产生了间接的左公因子, 因此如果不修改文法就不能使用递归下降分析法。

如何更精确地判断一个文法能用递归下降分析法?



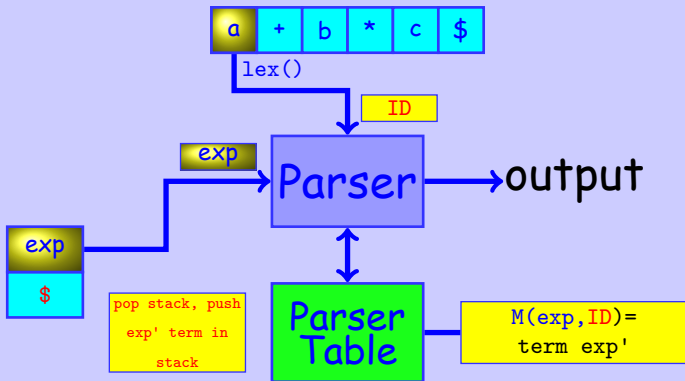
## 分析器的设计



对应的最左推导： $\exp \xRightarrow[\text{lm}]{*}$

已弹出的栈顶符号    栈内符号自顶向下排列

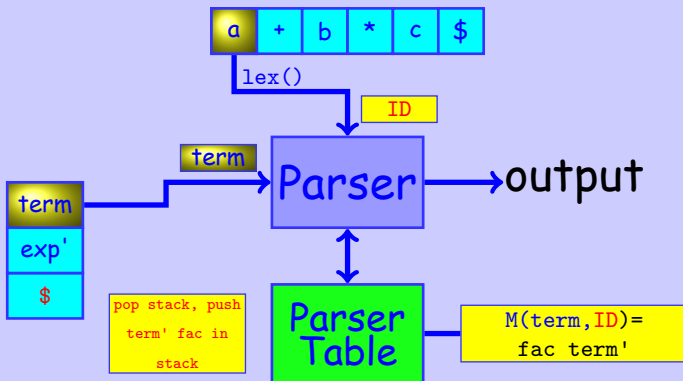
## 分析器的设计



对应的最左推导:  $\text{exp} \xRightarrow[\text{lm}]{*}$    $\text{exp}$

已弹出的栈顶符号 栈内符号自顶向下排列

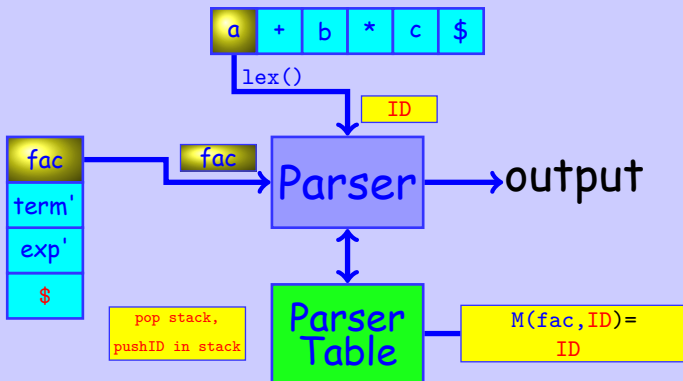
## 分析器的设计



对应的最左推导:  $exp \xRightarrow{*}_{lm}$   term exp'

已弹出的栈顶符号 栈内符号自顶向下排列

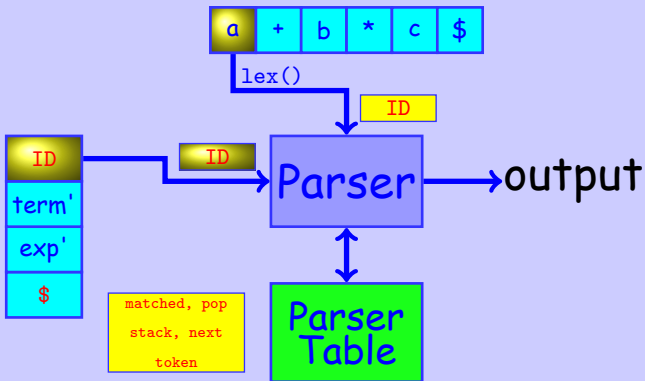
## 分析器的设计



对应的最左推导:  $exp \xRightarrow{*}_{lm}$  `fac term' exp'`

已弹出的栈顶符号 栈内符号自顶向下排列

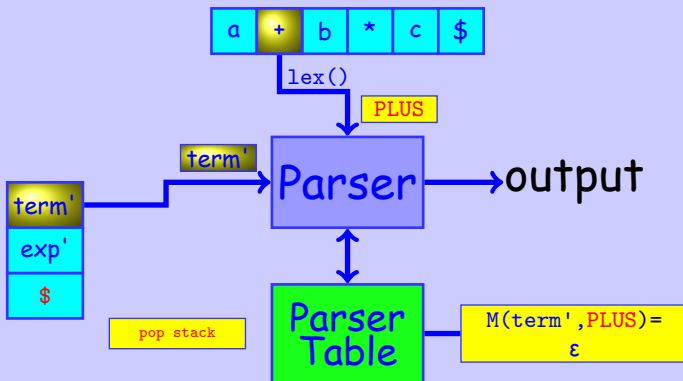
## 分析器的设计



对应的最左推导:  $exp \xRightarrow{*} \text{ID term' exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列

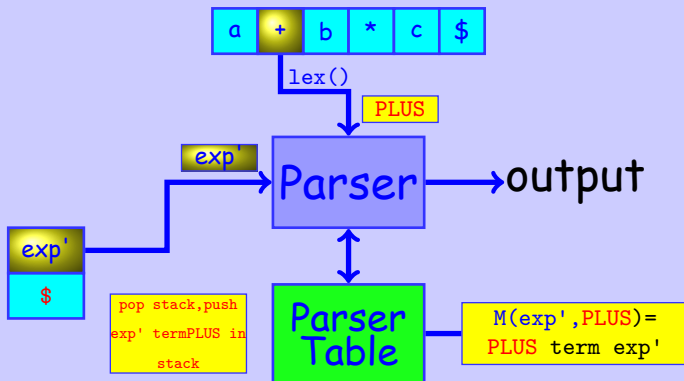
## 分析器的设计



对应的最左推导:  $exp \xRightarrow{*}_{lm} \text{ID term' exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列

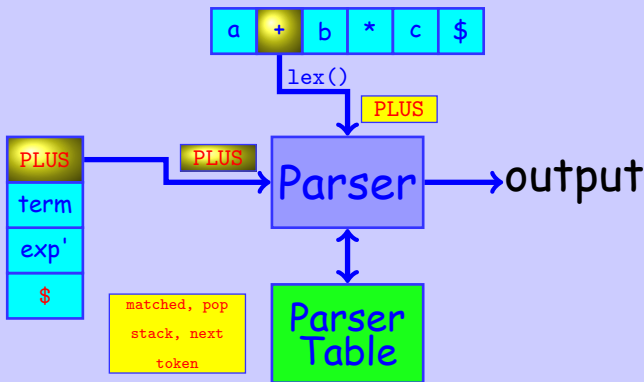
## 分析器的设计



对应的最左推导:  $exp \xRightarrow{*}_{lm} \text{ID } exp'$

已弹出的栈顶符号 栈内符号自顶向下排列

## 分析器的设计

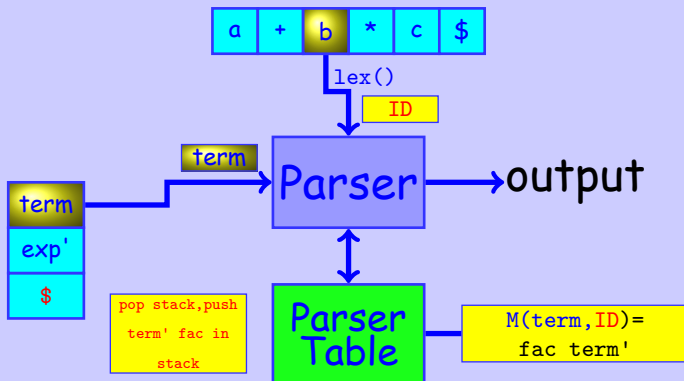


对应的最左推导:  $\text{exp} \xRightarrow{*} \text{ID PLUS term exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列



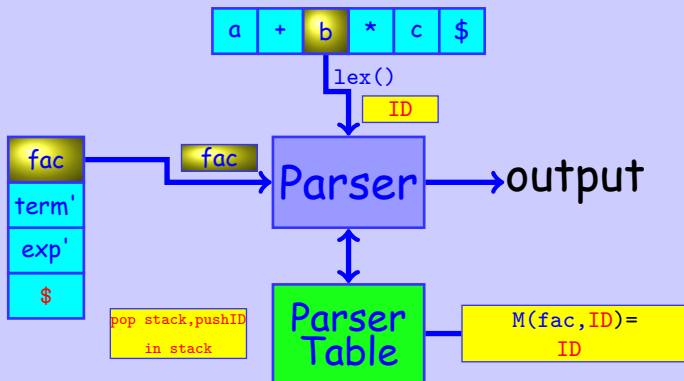
# 分析器的设计



对应的最左推导:  $exp \xRightarrow{*}_{lm} \text{ID PLUS term } exp'$

已弹出的栈顶符号 栈内符号自顶向下排列

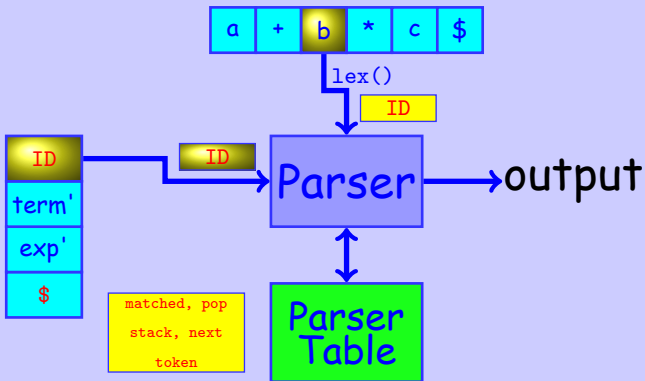
## 分析器的设计



对应的最左推导:  $\text{exp} \xRightarrow{*} \text{ID PLUS fac term' exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列

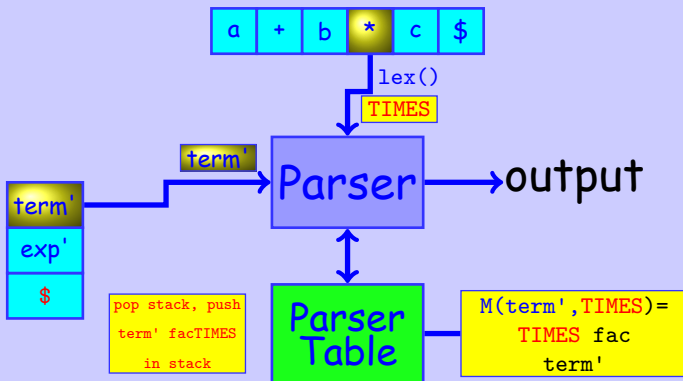
## 分析器的设计



对应的最左推导:  $exp \xRightarrow{*} ID \text{ PLUS } ID \text{ term' } exp'$

已弹出的栈顶符号 栈内符号自顶向下排列

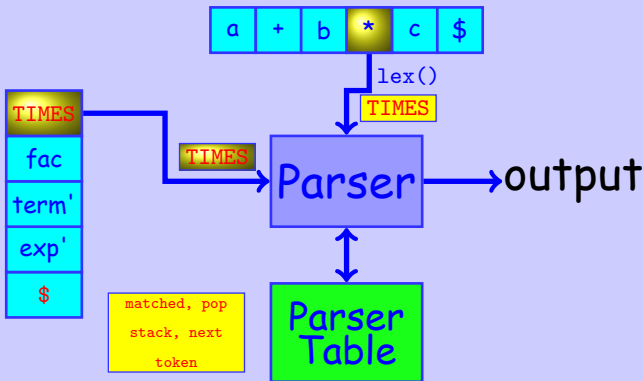
## 分析器的设计



对应的最左推导:  $exp \xRightarrow{*}_{lm} \text{ID PLUS ID term' exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列

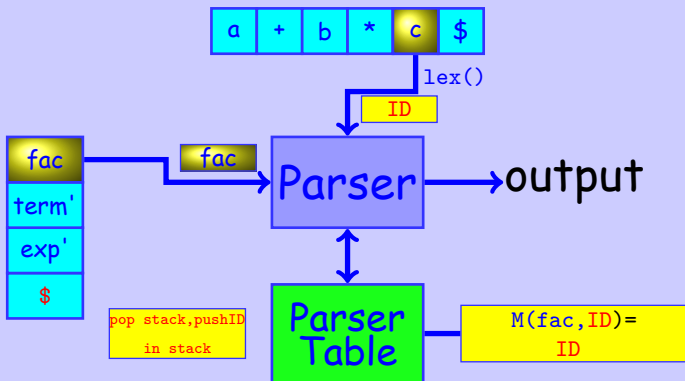
## 分析器的设计



对应的最左推导:  $\text{exp} \xrightarrow{*}_{\text{lm}} \text{ID PLUS ID TIMES fac term' exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列

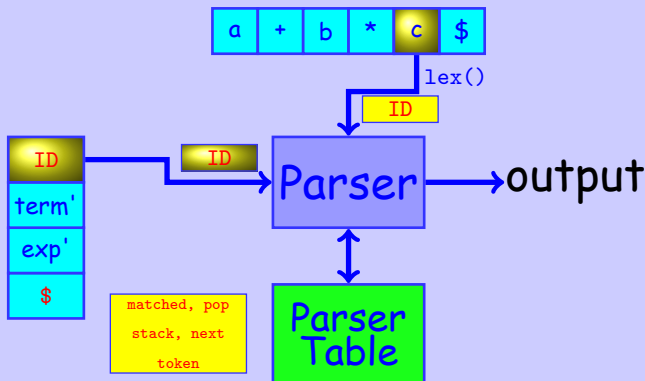
## 分析器的设计



对应的最左推导:  $exp \xRightarrow{*}_{lm} \text{ID PLUS ID TIMES fac term' exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列

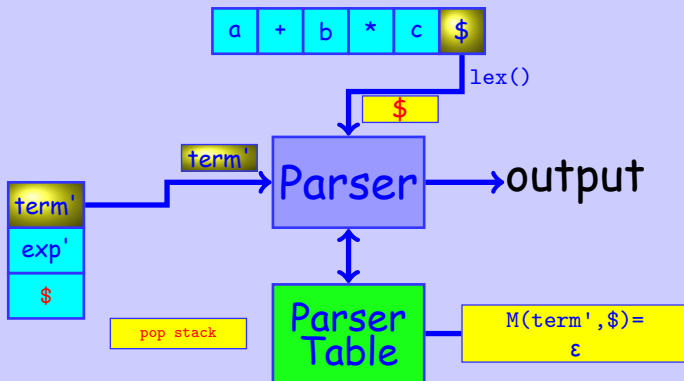
## 分析器的设计



对应的最左推导:  $\text{exp} \xrightarrow{*}_{\text{lm}} \text{ID PLUS ID TIMES ID term' exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列

## 分析器的设计

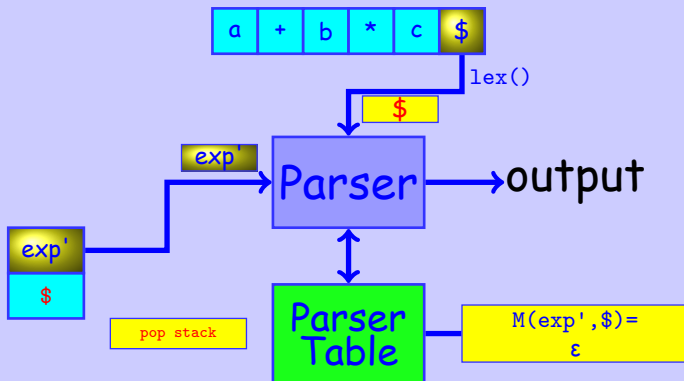


对应的最左推导:  $\text{exp} \xRightarrow{*}_{lm} \text{ID PLUS ID TIMES ID term' exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列



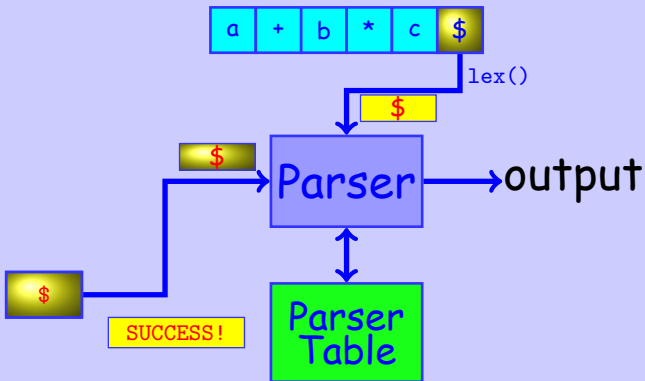
## 分析器的设计



对应的最左推导:  $\text{exp} \xRightarrow{*}_{lm} \text{ID PLUS ID TIMES ID exp'}$

已弹出的栈顶符号 栈内符号自顶向下排列



## 分析器的设计



对应的最左推导:  $\text{exp} \xrightarrow[\text{IM}]{*} \text{ID PLUS ID TIMES ID}$

已弹出的栈顶符号 栈内符号自顶向下排列

# Parser Action

- ① if (current\_top == current\_token && current\_top == \$)  
output success;
- ② if (current\_top == current\_token && current\_top != \$)  
pop stack; current\_token = getnext();
- ③ if (current\_top is non-terminal)  
if (M[current\_top][current\_token] == a)  
{ pop stack; 将 a 的符号串反向压入 stack; }  
else  
output error; 
- ④ if (current\_top is terminal &&  
current\_top != current\_token)  
output error; 

# Parsing Algorithm

```

Boolean ll_parse()
{
    symbol current_top = Start Symbol, current_token = getnext();
    Stack stack;
    symbol string a;
    push($); push(Start Symbol);
    do {
        if (current_top is terminal) {
            if (current_top == current_token) {
                pop(); current_top = top();
                current_token = getnext();
            } else
                error();
        } else
            if ( (a = M[current_top][current_input]) != error) {
                pop(); 将a反向压入栈中;
                current_top = top();
            } else
                error();
    } while (current_top != $);
    if (current_token == $) return True;
    else error();
}

```

## 分析表

## XL 文法

$$\text{exp} \rightarrow \text{term exp'}$$

$$| \epsilon$$

$$\text{exp'} \rightarrow + \text{term exp'}$$

$$| \epsilon$$

$$\text{term} \rightarrow \text{fac term'}$$

$$\text{term'} \rightarrow * \text{fac term'}$$

$$| \epsilon$$

$$\text{fac} \rightarrow \text{ID}$$

$$| ( \text{exp} )$$

## 分析表

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

## 分析过程

stack	input	action
\$exp	a+b*c\$	$exp \rightarrow term\ exp'$
\$exp' term	a+b*c\$	$term \rightarrow fac\ term'$
\$exp' term' fac	a+b*c\$	$fac \rightarrow ID$
\$exp' term' ID	a+b*c\$	match-advance
\$exp' term'	+b*c\$	$term' \rightarrow \epsilon$
\$exp'	+b*c\$	$exp' \rightarrow +\ term\ exp'$
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	$term \rightarrow fac\ term'$
\$exp' term' fac	b*c\$	$fac \rightarrow ID$
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	$term' \rightarrow *\ fac\ term'$
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	$fac \rightarrow ID$
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	$term' \rightarrow \epsilon$
\$exp'	\$	$exp' \rightarrow \epsilon$
\$	\$	success

oooooooo

oooooooooooo●

oooooooooooooooooooo

oooooooooooo

## 分析过程

stack	input	action
\$exp	a+b*c\$	exp $\rightarrow$ term exp'
\$exp' term	a+b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	a+b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	a+b*c\$	match-advance
\$exp' term'	+b*c\$	term' $\rightarrow$ $\epsilon$
\$exp'	+b*c\$	exp' $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success

oooooooo

oooooooooooo●

oooooooooooooooooooo

oooooooooooo

## 分析过程

stack	input	action
\$exp	a+b*c\$	exp $\rightarrow$ term exp'
\$exp' term	a+b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	a+b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	a+b*c\$	match-advance
\$exp' term'	+b*c\$	term' $\rightarrow$ $\epsilon$
\$exp'	+b*c\$	exp' $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success



## 分析过程

stack	input	action
\$exp	a+b*c\$	exp $\rightarrow$ term exp'
\$exp' term	a+b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	a+b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	a+b*c\$	match-advance
\$exp' term'	+b*c\$	term' $\rightarrow$ $\epsilon$
\$exp'	+b*c\$	exp' $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success

## 分析过程

stack	input	action
\$exp	a+b*c\$	exp $\rightarrow$ term exp'
\$exp' term	a+b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	a+b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	a+b*c\$	match-advance
\$exp' term'	+b*c\$	term' $\rightarrow$ $\epsilon$
\$exp'	+b*c\$	exp' $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success

## 分析过程

stack	input	action
\$exp	a+b*c\$	exp $\rightarrow$ term exp'
\$exp' term	a+b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	a+b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	a+b*c\$	match-advance
\$exp' term'	+b*c\$	term' $\rightarrow \epsilon$
\$exp'	+b*c\$	exp' $\rightarrow +$ term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp'	\$	exp' $\rightarrow \epsilon$
\$	\$	success

## 分析过程

stack	input	action
\$exp	a+b*c\$	exp $\rightarrow$ term exp'
\$exp' term	a+b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	a+b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	a+b*c\$	match-advance
\$exp' term'	+b*c\$	term' $\rightarrow \epsilon$
\$exp'	+b*c\$	exp' $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp'	\$	exp' $\rightarrow \epsilon$
\$	\$	success

## 分析过程

stack	input	action
\$exp	a+b*c\$	exp $\rightarrow$ term exp'
\$exp' term	a+b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	a+b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	a+b*c\$	match-advance
\$exp' term'	+b*c\$	term' $\rightarrow \epsilon$
\$exp'	+b*c\$	exp' $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp'	\$	exp' $\rightarrow \epsilon$
\$	\$	success

## 分析过程

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp	+D c\$	exp $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow$ $\epsilon$
\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success

## 分析过程

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp	+D c\$	exp $\rightarrow$ + term exp
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow$ $\epsilon$
\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success

## 分析过程

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp	+ID c\$	exp $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow$ $\epsilon$
\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success



## 分析过程

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp	+ID c\$	exp $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow$ $\epsilon$
\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success

## 分析过程

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp	+ID c\$	exp $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow$ $\epsilon$
\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success

## 分析过程

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp	+ID c\$	exp $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow \epsilon$
\$exp'	\$	exp' $\rightarrow \epsilon$
\$	\$	success

## 分析过程

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp	+ID c\$	exp $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow \epsilon$
\$exp'	\$	exp' $\rightarrow \epsilon$
\$	\$	success

## 分析过程

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp	+ID c\$	exp $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow$ $\epsilon$
\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success

## 分析过程

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

\$exp	+ID c\$	exp $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow \epsilon$
\$exp'	\$	exp' $\rightarrow \epsilon$
\$	\$	success

## 分析过程

stack	input	action
\$exp	a+b*c\$	exp $\rightarrow$ term exp'
\$exp' term	a+b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	a+b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	a+b*c\$	match-advance
\$exp' term'	+b*c\$	term' $\rightarrow$ $\epsilon$
\$exp'	+b*c\$	exp' $\rightarrow$ + term exp'
\$exp' term +	+b*c\$	match-advance
\$exp' term	b*c\$	term $\rightarrow$ fac term'
\$exp' term' fac	b*c\$	fac $\rightarrow$ ID
\$exp' term' ID	b*c\$	match-advance
\$exp' term'	*c\$	term' $\rightarrow$ * fac term'
\$exp' term' fac *	*c\$	match-advance
\$exp' term' fac	c\$	fac $\rightarrow$ ID
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	term' $\rightarrow$ $\epsilon$
\$exp'	\$	exp' $\rightarrow$ $\epsilon$
\$	\$	success

# 分析表的构造

## 定义

$$M[A][a] = A \rightarrow a \text{ iff } S \xRightarrow{*}_{lm} wAy \xRightarrow{*} wa\beta$$

## 两种情况

### 1 直接:

$$\left. \begin{array}{l} \Rightarrow \text{exp} \\ \Rightarrow \text{term exp}' \\ \Rightarrow \text{fac term}' \text{exp}' \\ \Rightarrow \text{ID term}' \text{exp}' \end{array} \right\} \Rightarrow M[\text{exp}][\text{ID}] = \text{'exp} \rightarrow \text{term exp}'$$

### 2 间接:

$$\left. \begin{array}{l} \Rightarrow \text{ID term}' \text{exp}' \\ \Rightarrow \text{ID exp}' \\ \Rightarrow \text{ID + term exp}' \end{array} \right\} \Rightarrow M[\text{term}'][+] = \text{'term}' \rightarrow \varepsilon'$$

不是 **term'** 直接展开生成 **+**, 而是句型中 **term'** 之后的文法符号串可以展开为 **+** 为首的文法符号串。



## 首符号集和后随符号集

## 定义

设  $\alpha$  是文法符号串， $A$  是非终结符：

$$\text{First}(\alpha) = \{a \mid a \in T, \alpha \xRightarrow{*} a\beta\} \cup \{\varepsilon \mid \text{if } \alpha \xRightarrow{*} \varepsilon\}$$

$$\text{Follow}(A) = \{a \mid a \in T, S \xRightarrow{*} \alpha A a \beta\} \cup \{\$ \mid \text{if } S \xRightarrow{*} \alpha A\}$$

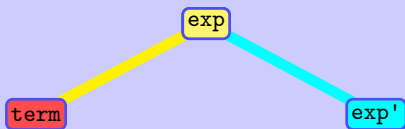
# 首符号和后随符号的传递性



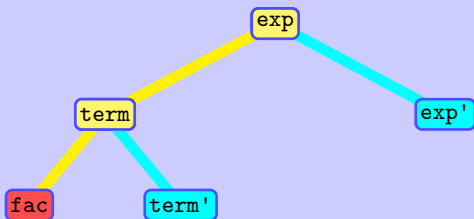
# 首符号和后随符号的传递性

exp

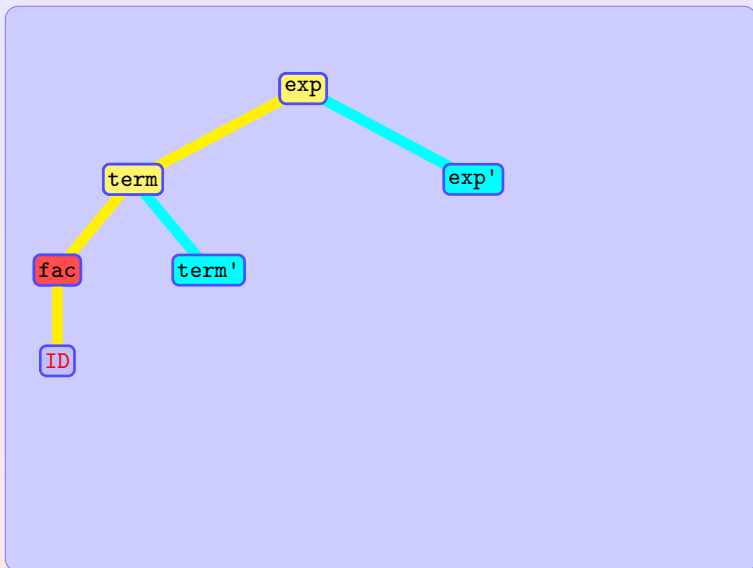
# 首符号和后随符号的传递性



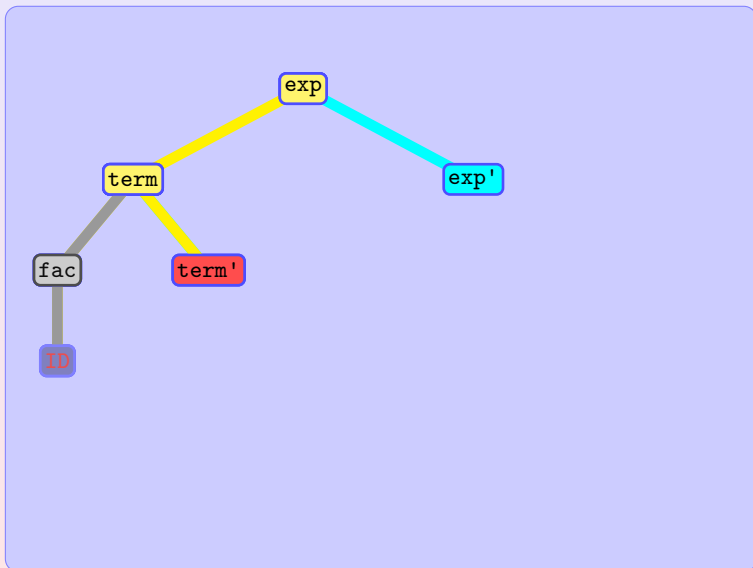
# 首符号和后随符号的传递性



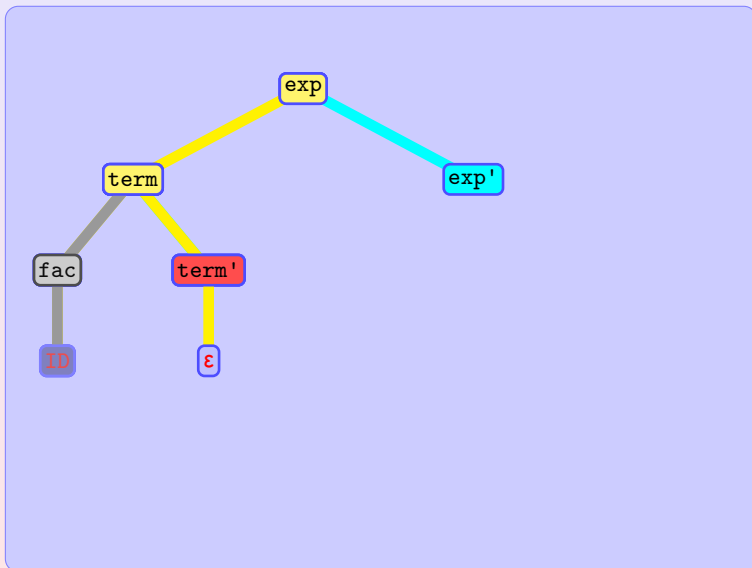
# 首符号和后随符号的传递性



# 首符号和后随符号的传递性

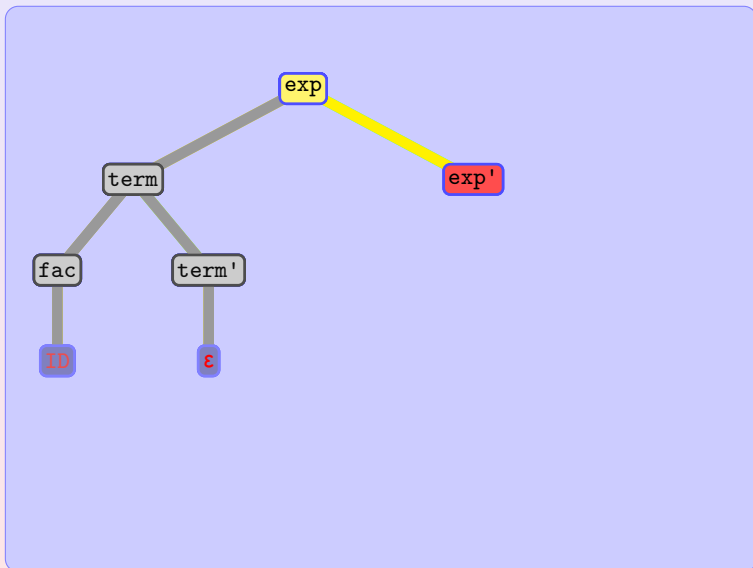


# 首符号和后随符号的传递性

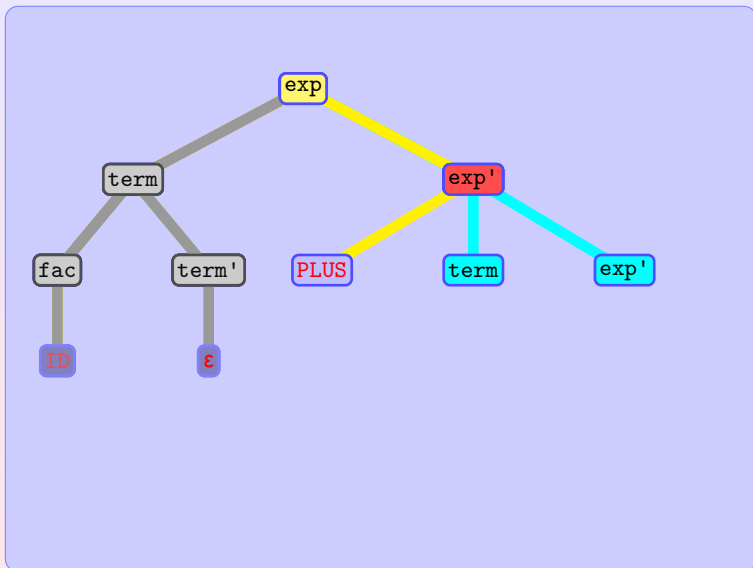




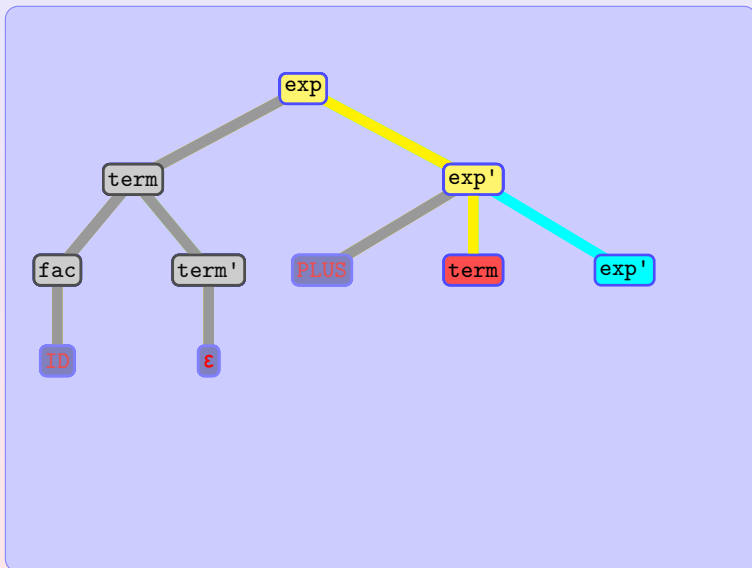
# 首符号和后随符号的传递性



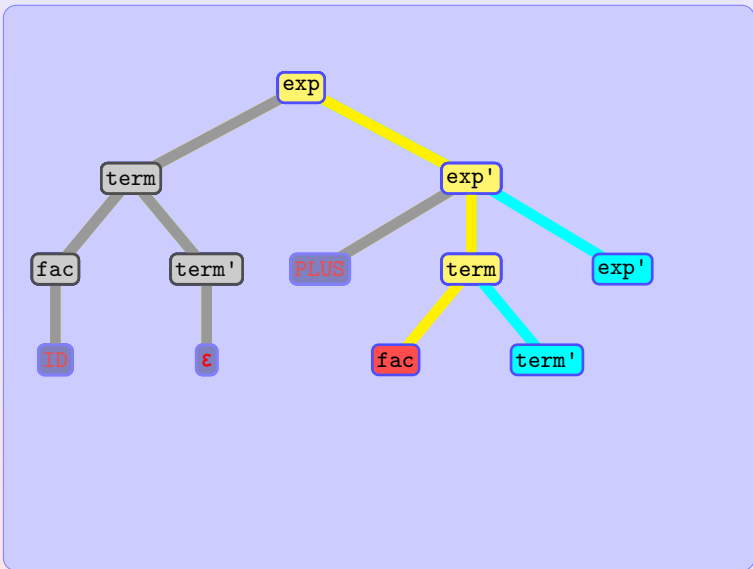
# 首符号和后随符号的传递性



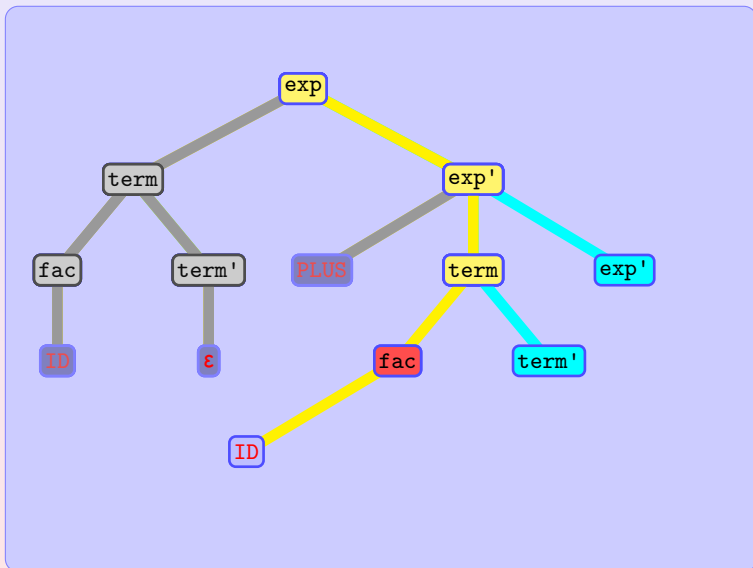
# 首符号和后随符号的传递性



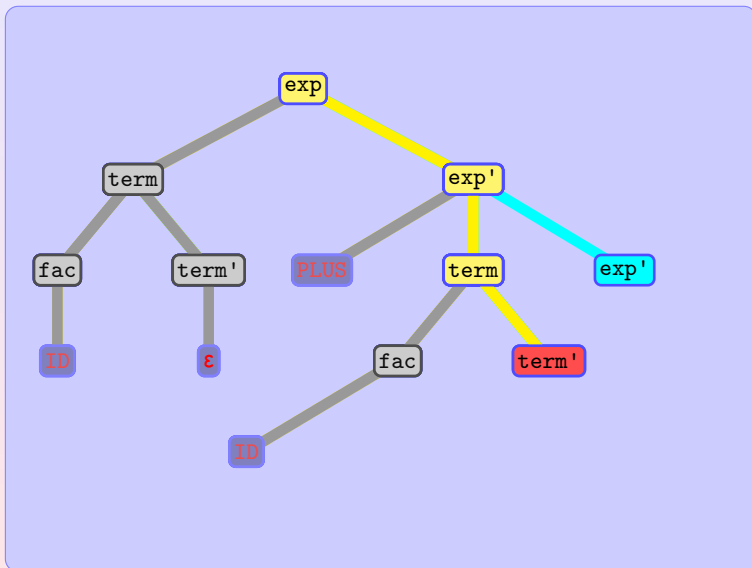
### 首符号和后随符号的传递性



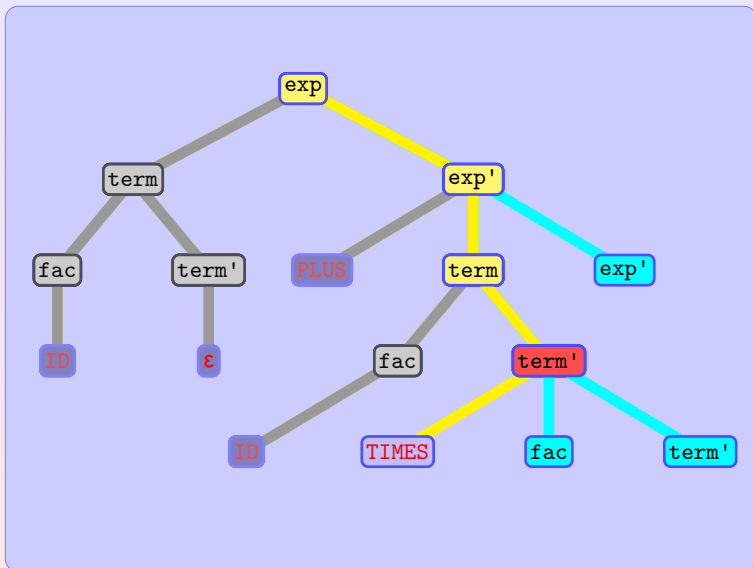
# 首符号和后随符号的传递性



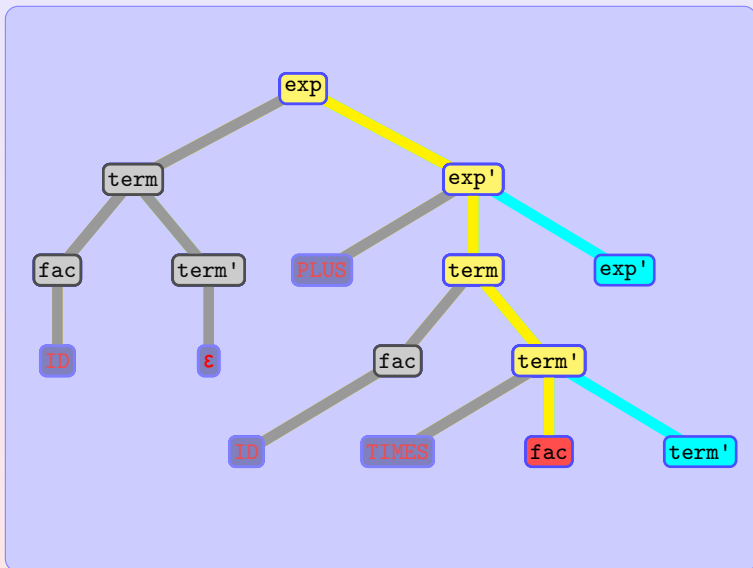
# 首符号和后随符号的传递性



# 首符号和后随符号的传递性

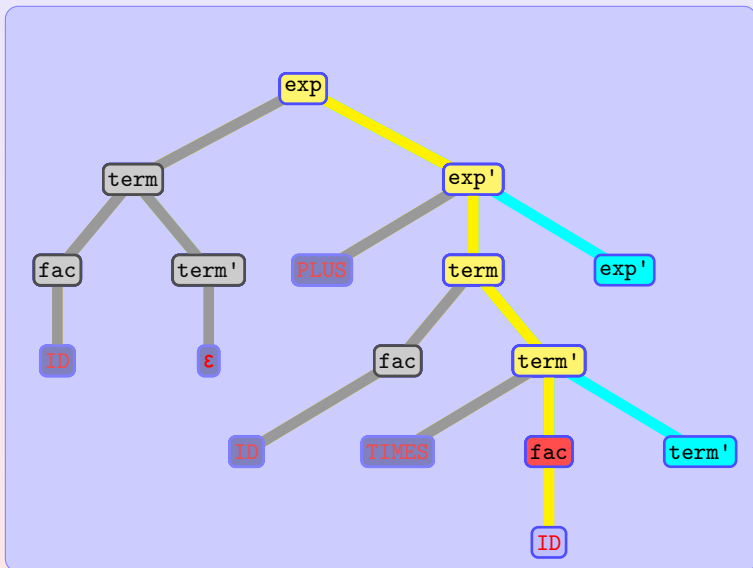


# 首符号和后随符号的传递性

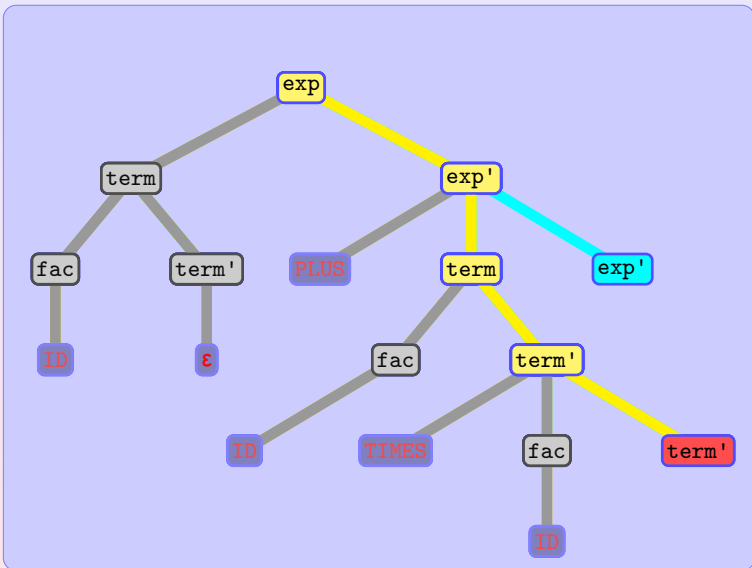




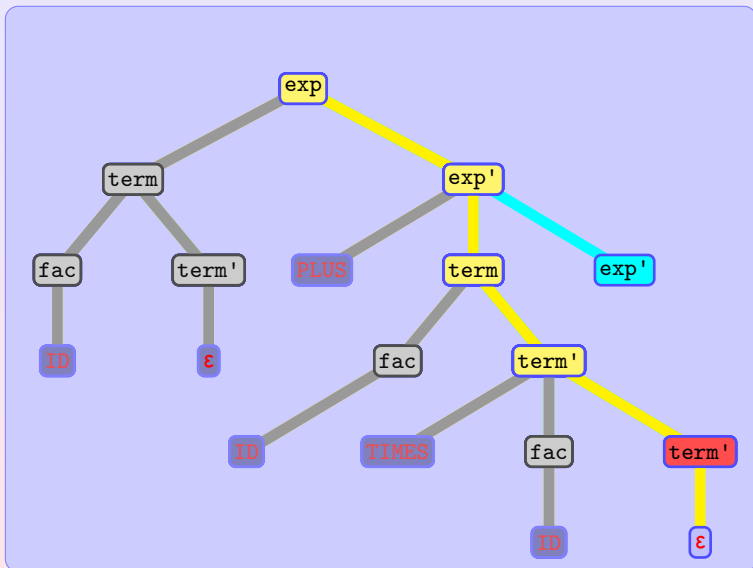
# 首符号和后随符号的传递性



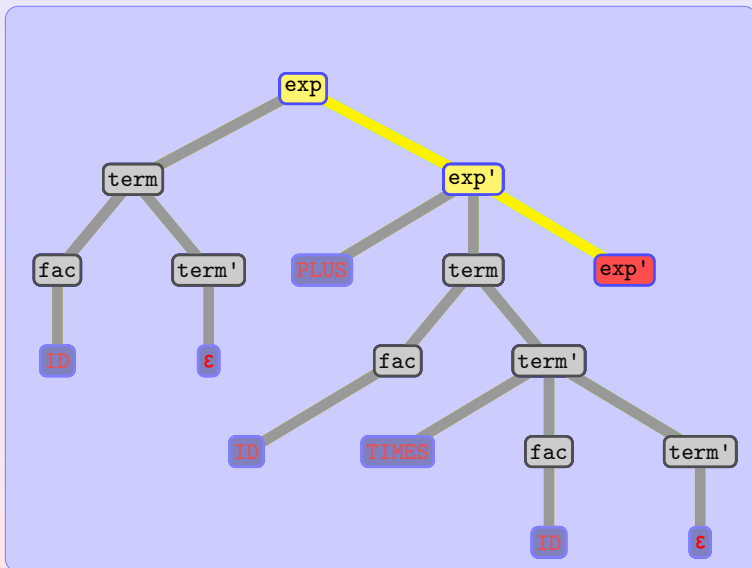
### 首符号和后随符号的传递性



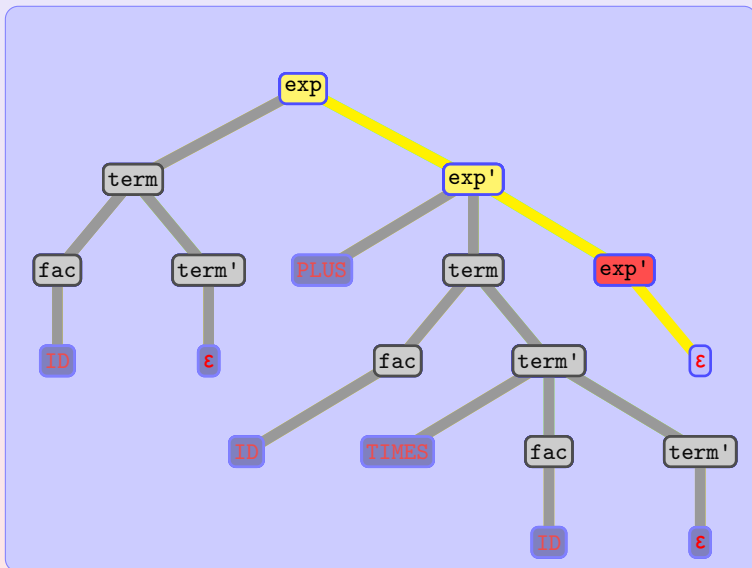
## 首符号和后随符号的传递性



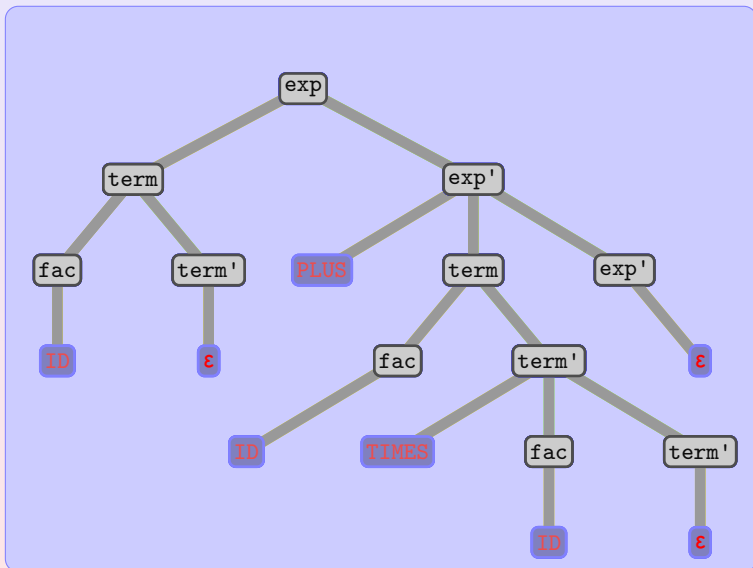
# 首符号和后随符号的传递性



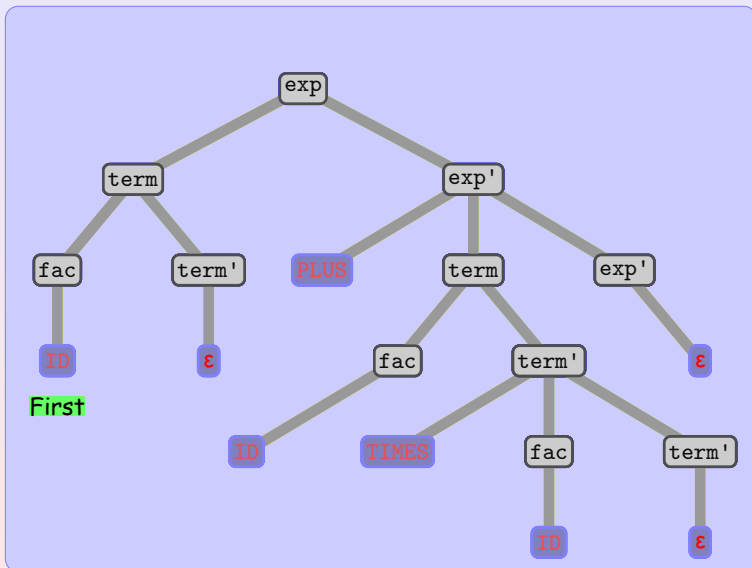
# 首符号和后随符号的传递性



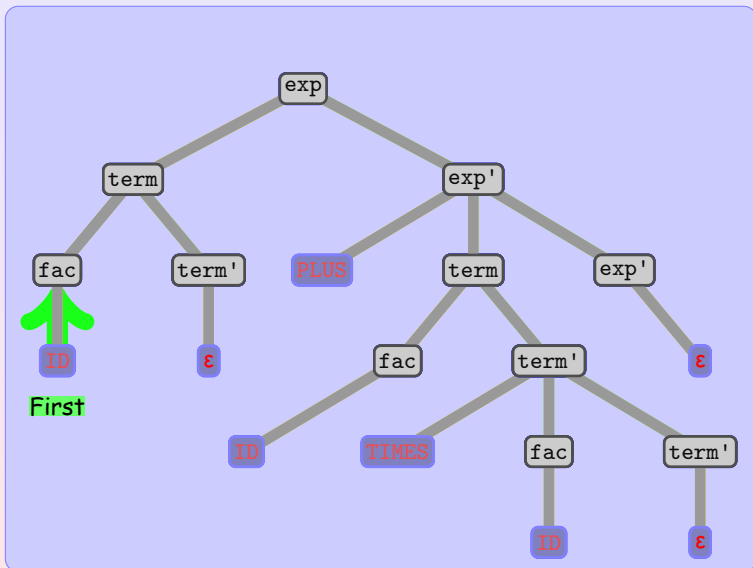
# 首符号和后随符号的传递性



# 首符号和后随符号的传递性

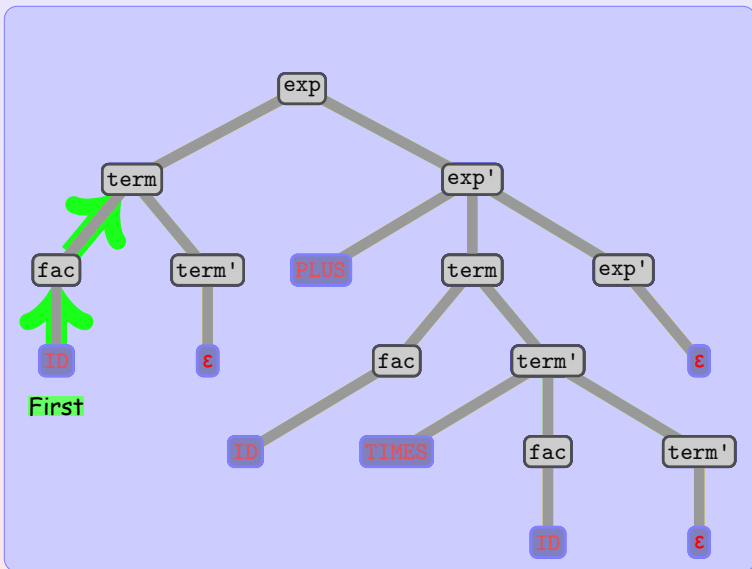


# 首符号和后随符号的传递性

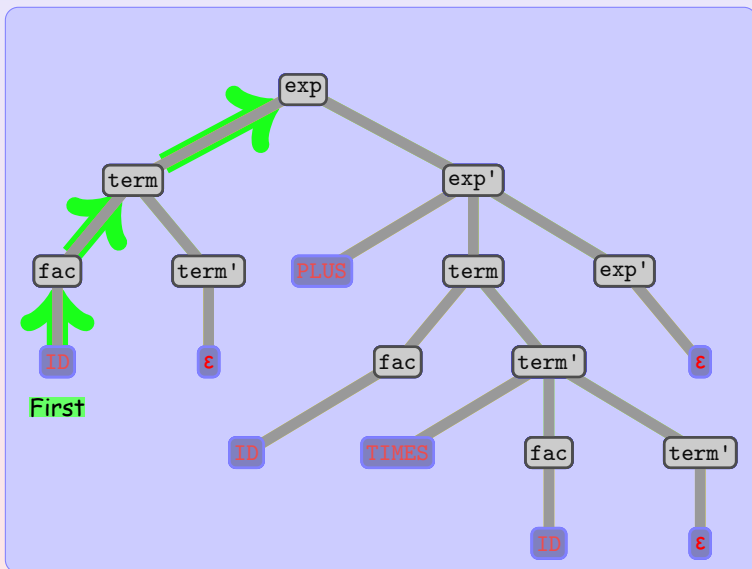




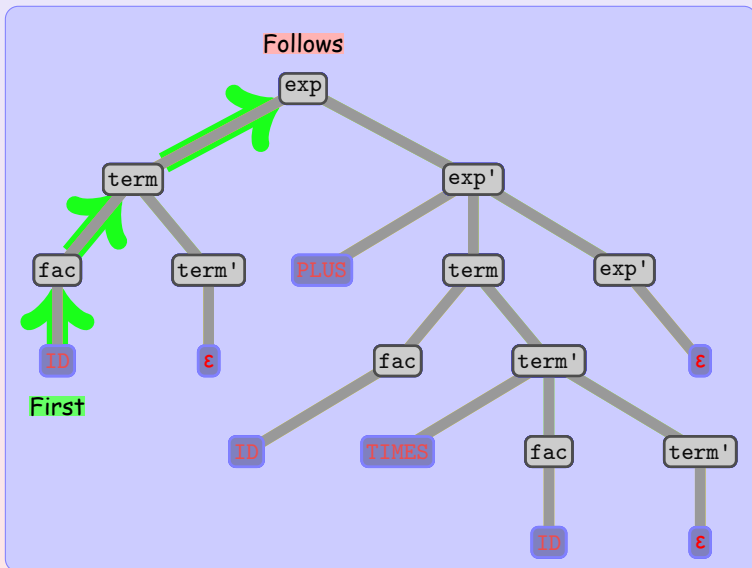
### 首符号和后随符号的传递性



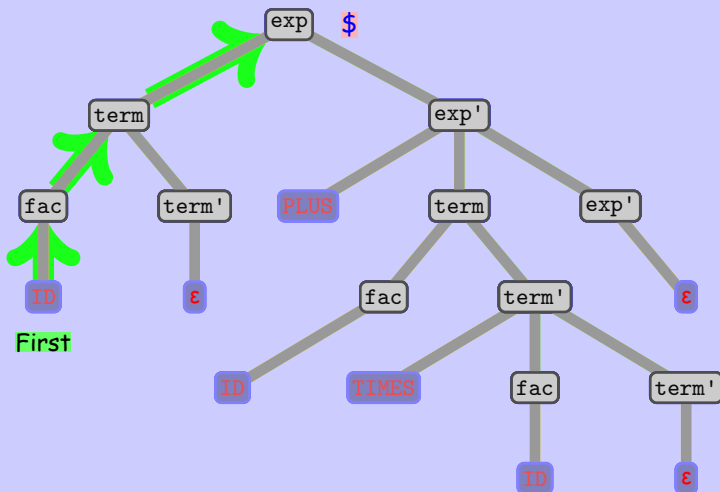
### 首符号和后随符号的传递性



# 首符号和后随符号的传递性



Follows



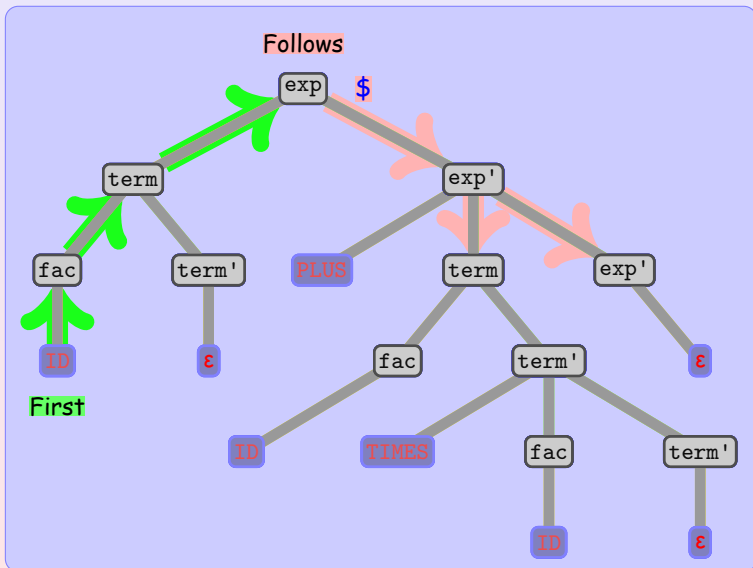
## Follows



Follows



# 首符号和后随符号的传递性

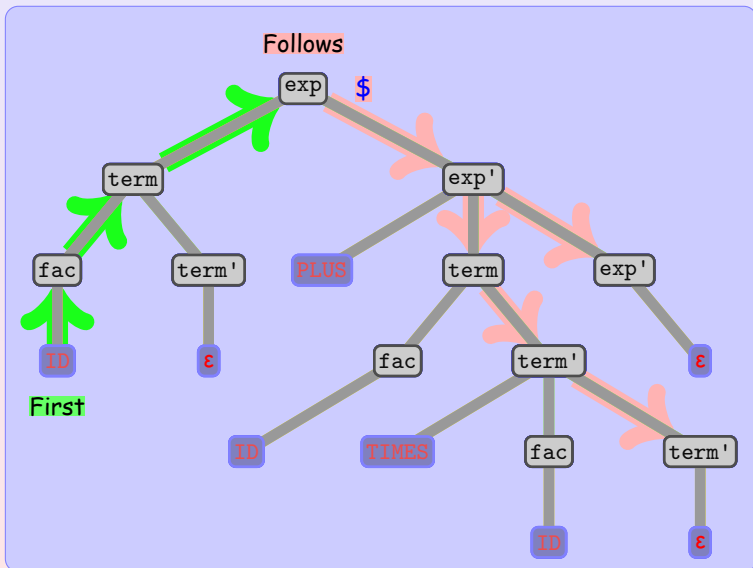


Follows

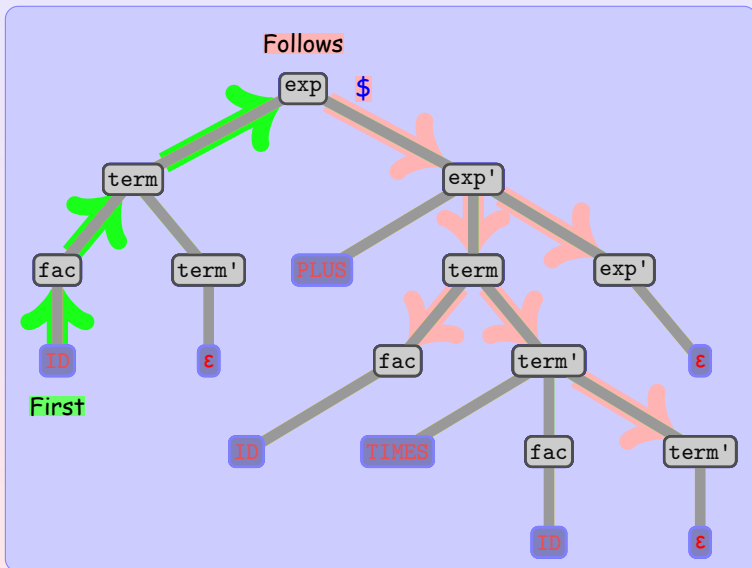




## 首符号和后随符号的传递性



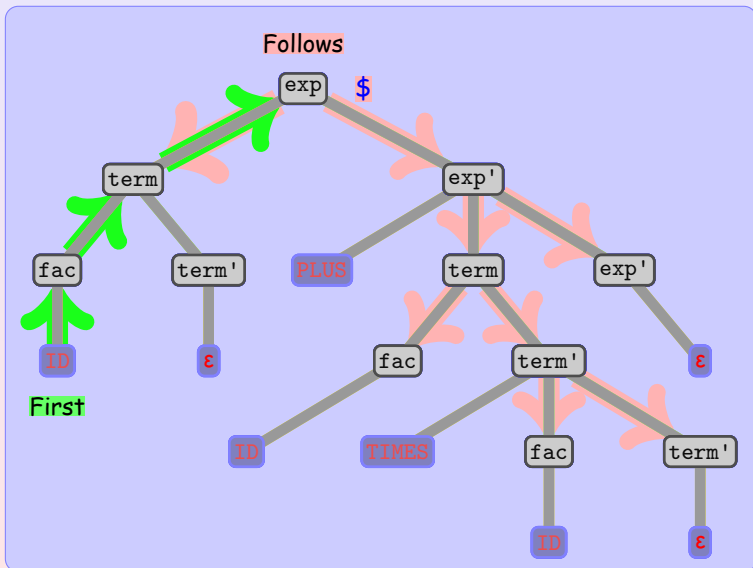
# 首符号和后随符号的传递性



## Follows



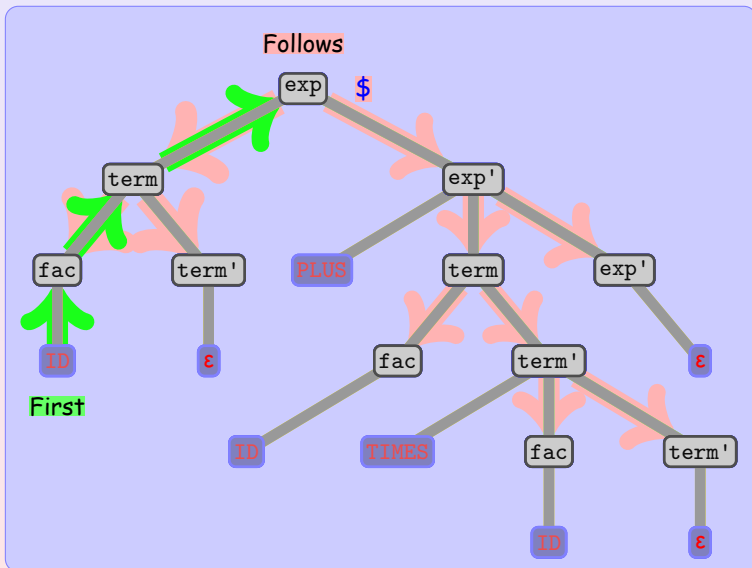
### 首符号和后随符号的传递性



## Follows



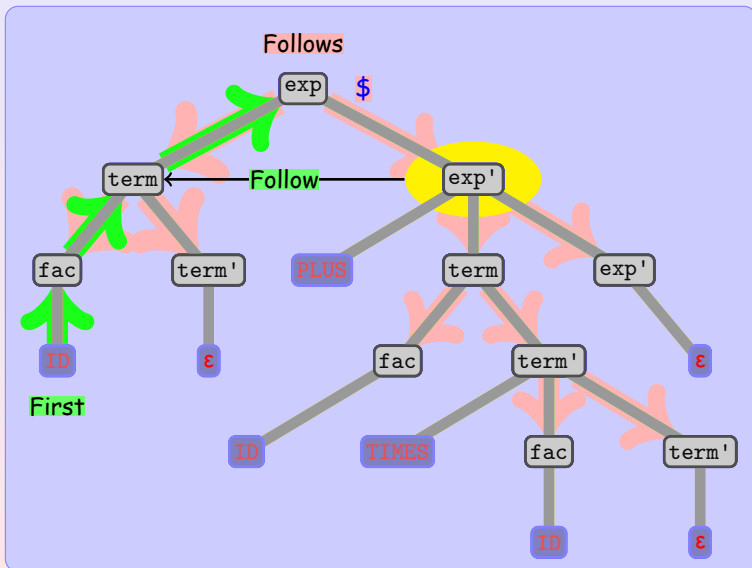
## 首符号和后随符号的传递性



## Follows



## 首符号和后随符号的传递性





# 首符号集的求解算法

## 递推规则

- 1 if  $X$  is terminal,  $\text{First}(X) = \{X\}$ ;
- 2 if  $X \rightarrow \varepsilon \in P, \varepsilon \in \text{First}(X)$ ;
- 3 if  $X \rightarrow Y_1 Y_2 \cdots Y_k \in P$ , then  
 $\text{First}(Y_1) \subseteq \text{First}(X)$   
 $\text{First}(Y_2) \subseteq \text{First}(X)$ ,  
 if  $\varepsilon \in \text{First}(Y_1)$   
 ...  
 $\text{First}(Y_k) \subseteq \text{First}(X)$ ,  
 if  $\varepsilon \in \bigcap_{i=1}^k \text{First}(Y_i)$

## 求解次序

先求文法层次较低非终结符，再求较高的。见

compiler\_cd/holub/src/compiler/  
parser/first.c.

## XL 文法

```
exp -> term exp'
exp' -> + term exp'
      | ε /* 空串 */
term -> fac term'
term' -> * fac term'
       | ε /* 空串 */
fac -> ID
      | ( exp )
```

## Example of XL

- $\text{First}(\text{fac}) = \{\text{ID}, (, \text{by } ③$
- $\text{First}(\text{term}') = \{\varepsilon, *\}$ , by ②③
- $\text{First}(\text{term}) = \text{First}(\text{fac})$ , by ③
- $\text{First}(\text{exp}') = \{\varepsilon, +\}$ , by ②③
- $\text{First}(\text{exp}) = \text{First}(\text{term})$ , by ③

# 后随符号集的求解算法

## 递归规则

- 1 if  $\$ \in \text{Follow}(S)$ ;
- 2 if  $A \rightarrow \alpha B \beta \in P$ ,  
 $\text{First}(\beta) - \{\epsilon\} \subseteq \text{Follow}(B)$ ;
- 3 if  $A \rightarrow \alpha B \in P$ , or  
 if  $A \rightarrow \alpha B \beta \in P$ , and,  $\beta \xRightarrow{*} \epsilon$   
 then  
 $\text{Follow}(A) \subseteq \text{Follow}(B)$   
 $\therefore$  if  $a \in \text{Follow}(A)$   
 $S \xRightarrow{*} \gamma A a \delta \xRightarrow{*} \gamma a B a \delta$

## 求解次序

先文法层次较高非终结符，再求较低的。见

compiler\_cd/holub/src/compiler/  
parser/follow.c.

## XL 文法

```
exp -> term exp'
exp' -> + term exp'
      | ε /* 空串 */
term -> fac term'
term' -> * fac term'
       | ε /* 空串 */
fac -> ID
      | ( exp )
```

## Example of XL

- $\text{Follow}(\text{exp}) = \{ \$, ) \}$ , by ①②
- $\text{Follow}(\text{exp}') = \text{Follow}(\text{exp})$ , by ③
- $\text{Follow}(\text{term}) = \{ \$, ), + \}$ , by ③②
- $\text{Follow}(\text{term}') = \text{Follow}(\text{term})$ , by ③
- $\text{Follow}(\text{fac}) = \{ \$, ), +, * \}$ , by ③②

## 求 select 集合

## select 集合的定义

$$M[A][a] = A \rightarrow a \iff S \xRightarrow[lm]{*} wAy \xRightarrow{*} wa\beta$$

$$\Rightarrow \begin{cases} \text{if } a \in \text{First}(A) \\ \text{if } a \xRightarrow{*} \epsilon, \text{ and } a \in \text{Follow}(A) \end{cases}$$

$$\iff a \in \text{First}(A) \cup \text{Follow}(A) \text{ if } a \xRightarrow{*} \epsilon$$

$$\iff a \in \text{Select}(A \rightarrow a)$$

## 必要条件并不是充分条件

$$S \rightarrow aSb \mid \epsilon; \text{Select}(S \rightarrow \epsilon) = \{b, \$\}$$

$S$  不能推出以  $b$  开始的句型，如果输入串是：“ $ba$ ”，分析器不是立刻出错，而是使用产生式  $S \rightarrow \epsilon$  将栈顶的  $S$  弹出后，栈中已经没有任何符号后才出错。



必要条件能保证如果输入是该语言的语句，则其对应的最左推导一定是分析器所推出的序列。但是不能保证能在第一时间发现错误，可能是在出现错误多步之后才报错。

## 首符号集

- $\text{First}(\text{fac}) = \{\text{ID}, (\}$
- $\text{First}(\text{term}') = \{\epsilon, *\}$
- $\text{First}(\text{term}) = \text{First}(\text{fac})$
- $\text{First}(\text{exp}') = \{\epsilon, +\}$
- $\text{First}(\text{exp}) = \text{First}(\text{term})$

## 后随符号集

- $\text{Follow}(\text{exp}) = \{\$, )\}$
- $\text{Follow}(\text{exp}') = \text{Follow}(\text{exp})$
- $\text{Follow}(\text{term}) = \{\$, ), +\}$
- $\text{Follow}(\text{term}') = \text{Follow}(\text{term})$
- $\text{Follow}(\text{fac}) = \{\$, ), +, *\}$

## Select 集合

- $\text{Select}(\text{exp} \rightarrow \text{term exp}') = \text{First}(\text{term}) = \{\text{ID}, (\}$
- $\text{Select}(\text{exp}' \rightarrow + \text{term exp}') = \{+\}$
- $\text{Select}(\text{exp}' \rightarrow \epsilon) = \text{Follow}(\text{exp}') = \{\$, )\}$
- $\text{Select}(\text{term} \rightarrow \text{fac term}') = \text{First}(\text{fac}) = \{\text{ID}, (\}$
- $\text{Select}(\text{term}' \rightarrow * \text{fac term}') = \{*\}$
- $\text{Select}(\text{term}' \rightarrow \epsilon) = \text{Follow}(\text{term}') = \{\$, ), +\}$
- $\text{Select}(\text{fac} \rightarrow \text{ID}) = \{\text{ID}\}$
- $\text{Select}(\text{fac} \rightarrow (\text{exp})) = \{( \}$

## 分析表

	ID	+	*	(	)	\$
exp	term exp'			term exp'		
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'			fac term'		
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID			( exp )		

# 分析表与递归下降分析法 1/2

## 分析表

	ID	+	*	(	)	\$
exp	term(); exp'();	term(); exp'();	term(); exp'();	term(); exp'();	term(); exp'();	term(); exp'();
exp'	return;	if ( match(+)){ advance(); term(); exp'();}	return;	return;	return;	return;
term	fac(); term'();	fac(); term'();	fac(); term'();	fac(); term'();	fac(); term'();	fac(); term'();
term'	return;	return;	if ( match(*)){ advance(); fac(); term'();}	return;	return;	return;
fac	if (match(ID)) advance();			if ( match(LP)){ advance(); exp(); if (match(RP)) advance(); else error();}		

在分析表中的出错栏上添加项目，将不会影响到分析器的识别能力，但是将会影响到分析器报错的时间。

## 分析表与递归下降分析法 2/2

### 更精确的分析

可以将exp对应的递归调用序列修改如下：

```
exp()
{
    if (match( ID ) || match( LP )){
        /* 注意：没有advance() */
        term ();
        exp' ()
    } else
        error();
}
```

分析器将及时发现错误，但是影响了识别的效率。

# 文法对分析栈的影响

## 含有右增长文法

$$S \rightarrow aSb \mid \varepsilon$$

分析器在分析语句  $a^n b^n$  的前  $n$  个  $a$  后, 栈的格局如下:

$$\$ \underbrace{bb \cdots b}_n S$$

表明栈有可能会无限增.

因此可能会导致内存资源的枯竭.

XL 语言也是如此:

$$\text{fac} \rightarrow (\text{exp}) \mid \text{ID}$$

# LL(1) 语言

## 冲突

- 冲突(conflict)

$$M[A][a] = \{A \rightarrow \alpha, A \rightarrow \beta\}$$

- 等价定义:

$$\text{Select}(A \rightarrow \alpha) \cap \text{Select}(A \rightarrow \beta) \neq \emptyset$$



## LL(1) 语言

- LL(1) 分析表中没有冲突项的文法称为 LL(1) 文法, 该文法生成的语言称为 LL(1) 语言.

## 定理

设  $G$  是 LL(1) 文法,  $p$  是文法  $G$  的 LL(1) 分析器, 则:  
 $w \in T^* \text{ iff } p(w) = \text{True}$



# LL(1) 语言

## 性质

- LL(1) 文法一定是无二义性的文法，二义文法也一定不是 LL(1) 文法。

## XL' 语言两个最左推导

$$\begin{aligned}
 \text{exp} &\xRightarrow{\text{lm}} \underline{\text{exp}} + \text{exp} \\
 &\xRightarrow{\text{lm}} \text{ID} + \underline{\text{exp}} \\
 &\xRightarrow{\text{lm}} \text{ID} + \underline{\text{exp}} * \text{exp} \\
 &\xRightarrow{\text{lm}} \text{ID} + \text{ID} * \underline{\text{exp}} \\
 &\xRightarrow{\text{lm}} \text{ID} + \text{ID} * \text{ID}
 \end{aligned}$$

$$\begin{aligned}
 \text{exp} &\xRightarrow{\text{lm}} \underline{\text{exp}} * \text{exp} \\
 &\xRightarrow{\text{lm}} \underline{\text{exp}} + \text{exp} * \text{exp} \\
 &\xRightarrow{\text{lm}} \text{ID} + \underline{\text{exp}} * \text{exp} \\
 &\xRightarrow{\text{lm}} \text{ID} + \text{ID} * \underline{\text{exp}} \\
 &\xRightarrow{\text{lm}} \text{ID} + \text{ID} * \text{ID}
 \end{aligned}$$

$$\text{ID} \in \text{Select}(\text{exp} \rightarrow \text{exp} * \text{exp}) \cap \text{Select}(\text{exp} \rightarrow \text{exp} + \text{exp})$$

## 选择冲突项解锁二义性

## if-then-else 结构

$$\begin{cases} S \rightarrow iE^+SS' \mid a \\ S' \rightarrow eS \mid \varepsilon \\ E \rightarrow b \end{cases}$$

$$\text{Select}(S' \rightarrow eS) \cap \text{Select}(S' \rightarrow \varepsilon) = \{e\}$$

- 如果, 选定  $M[S'][e] = S' \rightarrow eS$ , else 总是将匹配最近的 if.
- 如果, 选定  $M[S'][e] = S' \rightarrow \varepsilon$ , 则 else 成分永远不会被识别, 因此破坏了分析识别能力.
- 没有一般的规则来选择冲突项解锁二义性同时不破坏分析器的识别能力.

# 构成 LL(1) 文法的条件

对文法的每个产生式  $A \rightarrow \alpha \mid \beta$

- $\alpha$  和  $\beta$  没有左公因子.
- $\alpha$  和  $\beta$  最多只有一个能推出  $\epsilon$ .
- 如果  $\alpha \xRightarrow{*} \epsilon$ , 则  $\beta$  的首符号一定不在  $\text{Follow}(A)$  中.

# Example 1/2

## 非 LL(1) 文法

$$\begin{cases} S \rightarrow aAaacc \mid bAbbcc \\ A \rightarrow aA \mid bA \mid \varepsilon \end{cases}$$

- $\because a, b \in \text{Follow}(A)$ , so  $\text{Select}(A \rightarrow \varepsilon) \cap \text{Select}(A \rightarrow aA) \neq \emptyset$ .
- 如果 First 和 Follow 集是查看连续两个终结符, 则  $a(a|b) \in \text{First}_2(aA)$ ,  $aa, bb \in \text{Follow}_2(A)$ , so  $\text{Select}(A \rightarrow \varepsilon) \cap \text{Select}(A \rightarrow aA) \neq \emptyset$ . 不是 LL(2) 文法.
- 如果 First 和 Follow 集是查看连续三个终结符, 则  $aaa, abb, a(a|b)^2 \in \text{First}_3(aA)$ ,  $aac, bbc \in \text{Follow}_3(A)$ , so  $\text{Select}(A \rightarrow \varepsilon) \cap \text{Select}(A \rightarrow aA) = \emptyset$ . 同样  $\text{Select}(A \rightarrow \varepsilon) \cap \text{Select}(A \rightarrow bA) = \emptyset$ . 这样的文法称为 LL(3) 文法.
- 向前查看  $k$  个符号, 且 Select 集不冲突的文法称为 LL( $k$ ) 文法;
- 能否修改文法为 LL(1)?

## Example 2/2

### 非 LL 语言




$$L = \{a^n b^n \mid n \in \mathbf{N}\} \cup \{a^n c^n \mid n \in \mathbf{N}\}$$

生成  $L$  的无二义文法  $G$  如下:

$$\begin{cases} S \rightarrow B \mid C \\ B \rightarrow aBb \mid \varepsilon \\ C \rightarrow aCc \mid ac \end{cases}$$

- 为了保持  $a$  与  $b$ ,  $a$  与  $c$  的平衡, 产生式必须左右同时增长。
- $a, b$  的平衡与  $a, c$  平衡必须使用不同的非终结符。
- 因此无法消除上述文法的间接左公因子。
- 无论向前查看多少个符号  $k$  个符号都不能构造  $LL(k)$  文法;
- 非 LL 语言。

## LL 文法自动生成工具

- **LLgen**: provides a tool for generating an efficient recursive descent parser with no backtrack from an Extended Context Free syntax(EBNF)   
(<http://www.cs.vu.nl/~ceriel/LLgen.html>).
- **JavaCC**: Java Compiler Compiler, mixed lexical & parser analyzer generator. Output Languages: Java, C/C++.  
(<https://javacc.java.net/>) 
- **ANTLR**: ANOther Tool for Language Recognition (ANTLR) is the parser generator based on LL(\*) parsing. successor to the Purdue Compiler Construction Tool Set (PCCTS). ANTLR supports generating code in the following languages: C, C++, Java, Python, C#, Objective-C, distributes as Java API  
(<http://www.antlr.org/>). 
- A. Holub **LLpar**: See CD-ROM.

## LL(1) 分析法出错处理

### 程序出错的统计

- 语法错误: Compiler 处理, 80% 的语法错误是标点错误.
- 语义错误: 部分可以处理 (类型错误), 部分需要用户 Debug.

### XL 实例

- $a * + b;$  (缺少运算量或多余运算符号)
- $a + b\_c) * d;$  (缺少运算量或多余运算符号, 括号不平衡)

### LL(1) 分析器


- input token and top token not match.
- $M[\text{top non-terminal}][\text{input token}] = \text{error}.$

### Parser 的任务

- 精确定位.
- 恢复错误继续分析.

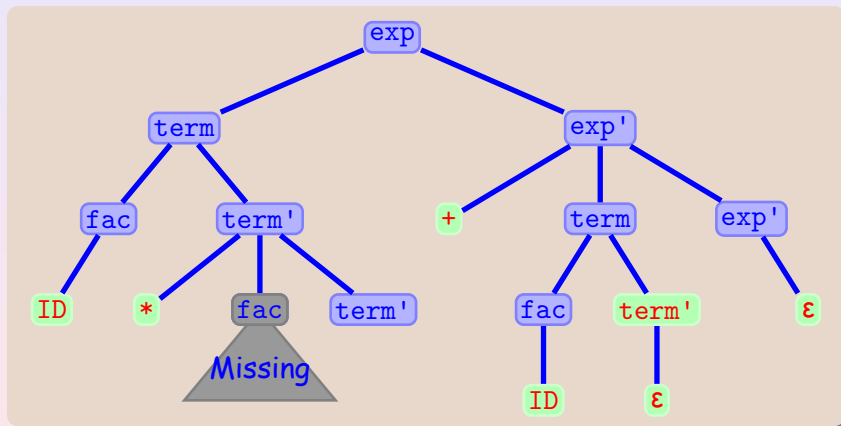
## 分析出错

stack	input	action
\$exp	a*+b\$	expand
...	...	...
\$exp' term'	*+b\$	expand
\$exp' term' fac *	*+b\$	match-advance
<b>\$exp' term' fac</b>	<b>+b\$</b>	<b>error and pop</b>
\$exp' term'	+b\$	expand
\$exp'	+b\$	expand
\$exp' term +	+b\$	expand
\$exp' term	b\$	match-advance
\$exp' term' fac	b\$	expand
\$exp' term' <b>ID</b>	b\$	match-advance
\$exp' term'	\$	expand
\$exp'	\$	expand
\$	\$	success

 **fac** 希望获得 **ID** 或 **(, +** 是 **fac** 的后随符号, **pop fac** 意味着已经识别了一个 **fac** 的语句成分.



# 出错分析所建立的语法树



## 同步符号的处理

## Example of XL

Configuration in Error Time:

stack	input
\$exp' term' fac	+b\$

$$M[fac][+] = \text{err}; + \in \text{Follow}(fac);$$
Parser 认为 Input 漏掉了  $fac$  的成分 “ID”

1	exp	$\xRightarrow{*}_{lm}$	ID * fac term' exp'
2		$\xRightarrow{*}_{lm}$	ID * ID term' exp'
3		$\xRightarrow{*}_{lm}$	ID * ID $\epsilon$ exp'

pop  $fac$  之后的 Configuration:

stack	input
\$exp' term'	+b\$

相当于跳过了推导②:

 $+ \in \text{Follow}(fac)$  称为  $fac$  的同步符号。

## 一般情况

Configuration in Error Time:

stack	input
$\beta^R A$	ayyyy\$

$$M[A][a] = \text{err}; a \in \text{Follow}(A)$$
Parser 认为 Input 漏掉了  $A$  的成分 “XXX”

1	S	$\xRightarrow{*}_{lm}$	w A $\beta$
2		$\xRightarrow{*}_{lm}$	w XXX $\beta$
3		$\xRightarrow{*}_{lm}$	w XXX a exp'

pop  $A$  之后的 Configuration:

stack	input
$\beta^R$	ayyyy\$

相当于跳过了推导②:

 $a \in \text{Follow}(A)$  称为  $A$  的同步符号。

## 分析表

	ID	+	*	(	)	\$
exp	term exp'			term exp'	synch	synch
exp'		+ term exp'			$\epsilon$	$\epsilon$
term	fac term'	synch		fac term'	synch	synch
term'		$\epsilon$	* fac term'		$\epsilon$	$\epsilon$
fac	ID	synch	synch	( exp )	synch	synch

## 非同步符号的处理

stack	input	action
\$exp	a b + c\$	expand
...	...	...
\$exp' term'	b+c\$	error-skip
\$exp' term'	+c\$	expand
\$exp' term +	+c\$	expand
\$exp' term	c\$	match-advance
\$exp' term' fac	c\$	expand
\$exp' term' ID	c\$	match-advance
\$exp' term'	\$	expand
\$exp'	\$	expand
\$	\$	success

term' 希望获得 \*, + 或 \$, 但是当前的输入 ID, 分析出错, ID  $\notin$  Follow(term'), 分析器恢复分析的动作是报错并跳过 ID, 继续读下一个单词, 直到出现分析表能够使分析器继续动作的符号或同步符号.

## 问题

## Example

文法片断:

$$\begin{cases} \text{stmt} \rightarrow \text{exp}; \\ \text{stmt} \rightarrow \text{if}(\text{exp}) \text{stmt} \end{cases}$$

错误程序片断:

$$\begin{cases} \text{a} = \text{b} + \text{c} \square \\ \text{if}(\text{a}) \text{c} = \text{b}; \end{cases}$$

第一个 **exp** 希望获得同步符号 “;”，但是，分析器将 skip 掉 “if (a) c = b” 才能恢复分析，**skip 的成分太多!**

## 解决方法

keyword **if** 的语法层次比 **exp** 要低，是 **stmt** 的同步符号，可以将之作为 **exp** 的同步符号，使分析器及时恢复分析。

## 小结 (1/2)

### LL(1) 分析表的构造步骤

- 文法转换：消除左递归和左公因子。
- 对每个非终结符求 **First** 和 **Follow** 集合。
- 对每个产生式求 **Select** 集合。
- 对同一个非终结符的不同产生式查看 **Select** 集合是否有冲突。
- 如果没有，填写分析表，注意文件结束标志 **\$** 也是分析表的一项。
- 出错处理：同步符号。

## 小结 (2/2)

### LL(1) 文法的特点

- 直观, 易于理解, 实现容易, 分析表体积小.
- 出错处理简单.
- 文法变换后破坏了原文法的完整性, 不利于以后的语义分析. 但现代的 LL 分析器自动生成工具使用 Extended Backus-Naur Form, 这样消除左递归后不破坏语法结构的完整性. 如:

●  $exp \rightarrow term(+term)^*$

因此 LL 分析法也开始受到欢迎, 如 Princeton Univ. lcc 就是用 Topdown

Parser(<http://www.cs.princeton.edu/software/lcc/>), ANTLR 也广泛使用.

- 不易于二义性的解消.
- 并不是所有无二义的上下文无关文法都有 LL(k) 文法.

## 本章小节

### 1 基本概念

- 引言
- 自顶向下的语法分析
- 文法的等价变换

### 2 预测分析法

- 基本概念
- 递归下降分析法
- LL(1) 分析法

### 3 LL(1) 文法

- 首符号集和后随符号集
- LL(1) 分析表的构造
- LL(1) 语言

### 4 LL(1) 分析法出错处理

- 相关概念
- 同步符号
- 非同步符号的处理