编译原理

3. 语法分析-自顶向下

rainoftime.github.io

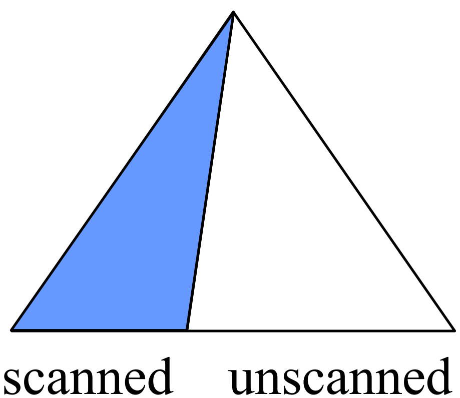
浙江大学
计算机科学与技术学院

课程内容

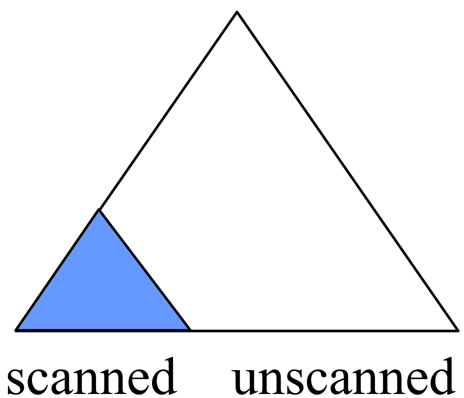
1. Introduction
2. Lexical Analysis
3. **Parsing**
4. Abstract Syntax
5. Semantic Analysis
6. Activation Record
7. Translating into Intermediate Code
8. Basic Blocks and Traces
9. Instruction Selection
10. Liveness Analysis
11. Register Allocation
13. Garbage Collection
14. Object-oriented Languages
18. Loop Optimizations

语法分析的主要方法

- **自顶向下(Top-down)**
  - 从文法的开始符号出发，尝试推导 (derive)出输入串
  - 分析树的构造方法：从根部开始
- **自底向上(Bottom-up)**
  - 尝试根据产生式规则归约(reduce)到文法的开始符号S
  - 分析树的构造方法: 从叶子开始



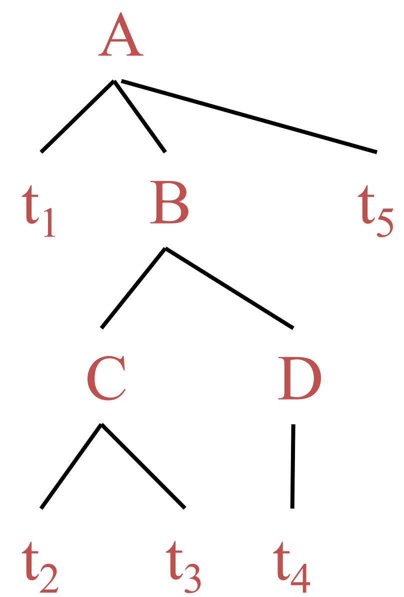
Top-down



Bottom-up

自顶向下语法分析

- **从文法开始符号*S*推导出串*w***
  - **E.g.,** $t_1\ t_2\ t_3\ t_4\ t_5$
  - 从顶部向底部方向构造Parse Tree
    - 从上往下、从左往右



- **每一步推导中，都需要做两个选择**
  1. 替换当前句型中的哪个非终结符?
  2. 用该非终结符的哪个产生式进行替换?

自顶向下语法分析

- **从分析树的顶部向底部方向构造分析树**
  - 从文法开始符号S推导出串w

- **每一步推导中，都需要做两个选择**
  1. 替换当前句型中的哪个非终结符?

自顶向下分析总是选择每个句型的最左非终结符进行替换!

  2. 用该非终结符的哪个产生式进行替换?

文法 $S \to cAd$

$A \to ab \mid b$

A有2个可选的产生式

本讲内容

1 **允许回溯的递归下降分析**

2 **LL(1)分析: 无回溯的自顶向下分析**

- LL(1)文法
- 递归实现
- 非递归实现

1. 允许回溯的递归下降分析

自顶向下分析的通用形式: 递归下降分析

- **递归下降分析(Recursive-Descent Parsing):**
  - 由一组过程/函数组成，每个过程对应一个非终结符
  - 从开始符号S对应的过程开始，(递归)调用其它过程
  - 如果S对应的过程恰好扫描了整个输入串，则成功完成分析

文法 $S \rightarrow cAd$

$A \rightarrow ab \mid b$

如何为S和A构造其对应的过程?

自顶向下分析的通用形式: 递归下降分析

- **递归下降分析(Recursive-Descent Parsing):**
  - 利用非终结符A的文法规则$A \rightarrow X_1 \ldots X_k$，定义识别$A$的过程
    - **如果X_i是非终结符**：调用相应非终结符对应的过程
    - **如果X_i是终结符a**：匹配输入串中对应的终结符a

```
    void A( ) {
1)     选择一个A产生式，A → X1 X2 ... Xk ;
2)     for ( i = 1 to k ) {
3)        if ( Xi是一个非终结符号)
4)           调用过程 Xi( ) ;
5)        else  if ( Xi 等于当前的输入符号a)
6)           读入下一个输入符号;
7)        else /* 发生了一个错误 */ ;
8)              ? ? ?
       }
    }
```

如果选择了不合适的产生式，可能需要**回溯**

例: 需要回溯的文法

考虑文法$S \rightarrow cAd$

$A \rightarrow ab \mid a$

为输入串$w = cad$ 建立分析树(假设按顺序选产生式)

cad $\boldsymbol{S}$

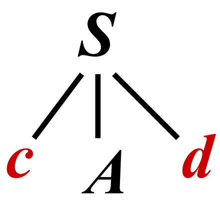
例: 需要回溯的文法

考虑文法$S \to cAd$

$A \to ab \mid a$

为输入串$w = cad$ 建立分析树(假设按顺序选产生式)

cad



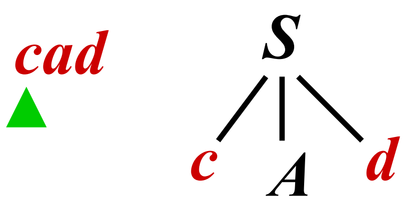
- **选择S的产生式(只有1个选择)**

例: 需要回溯的文法

考虑文法 $S \to cAd$

$A \to ab \mid a$

为输入串 $w = cad$ 建立分析树(假设按顺序选产生式)



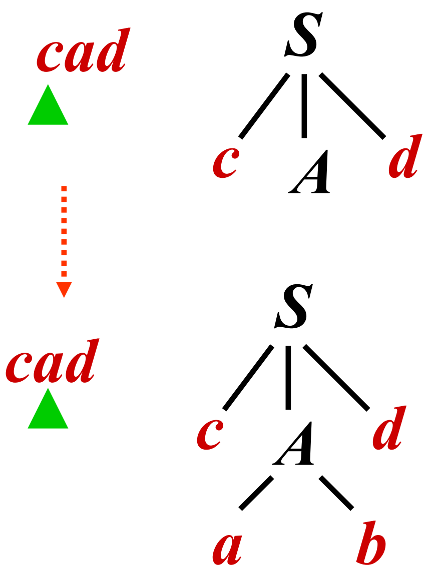
- **选择S的产生式(只有1个选择)**

例: 需要回溯的文法

考虑文法$S \rightarrow cAd$

$A \rightarrow ab \mid a$

为输入串$w = cad$ 建立分析树(假设按顺序选产生式)



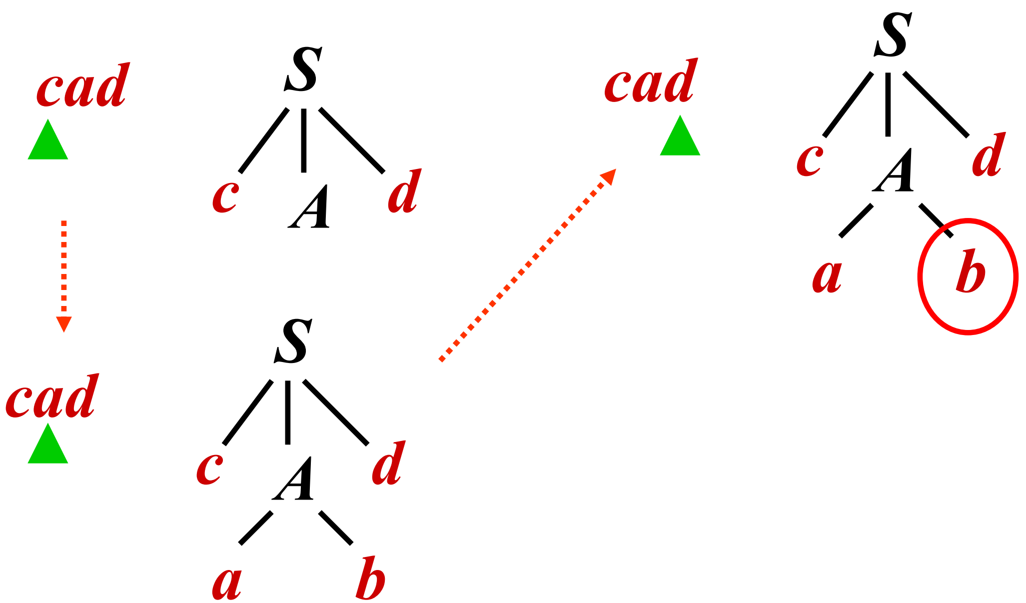
- **选择A的第1个产生式**$A \rightarrow ab$

例: 需要回溯的文法

考虑文法$S \rightarrow cAd$

$A \rightarrow ab \mid a$

为输入串$w = cad$ 建立分析树(假设按顺序选产生式)



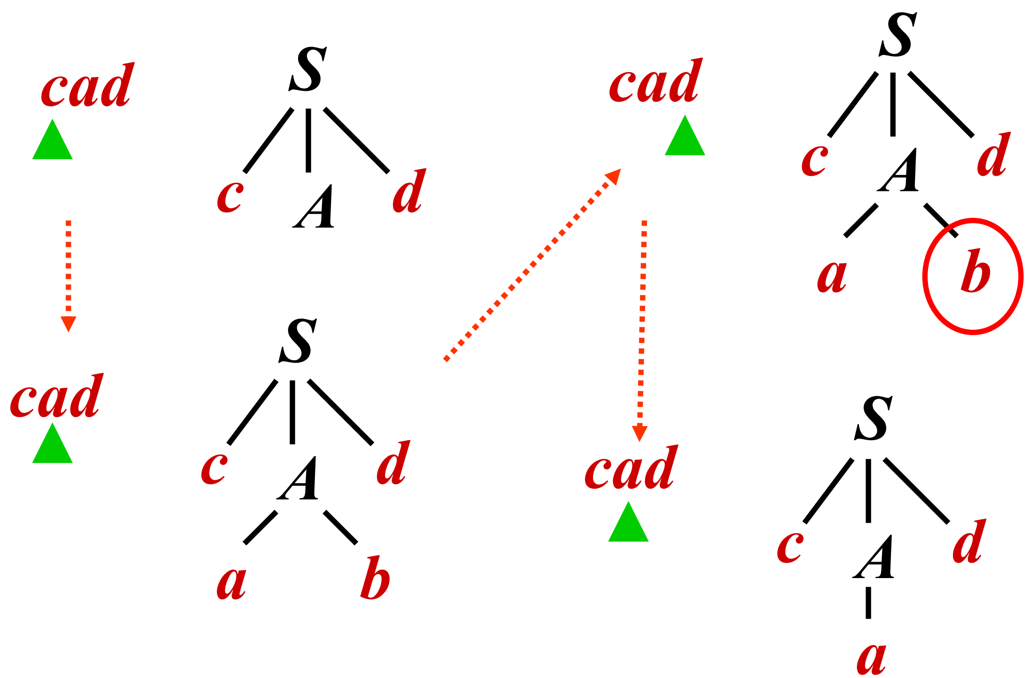
出现问题: 需要回溯

例: 需要回溯的文法

考虑文法$S \rightarrow cAd$

$A \rightarrow ab \mid a$

为输入串$w = cad$ 建立分析树(假设按顺序选产生式)



出现问题: 需要回溯

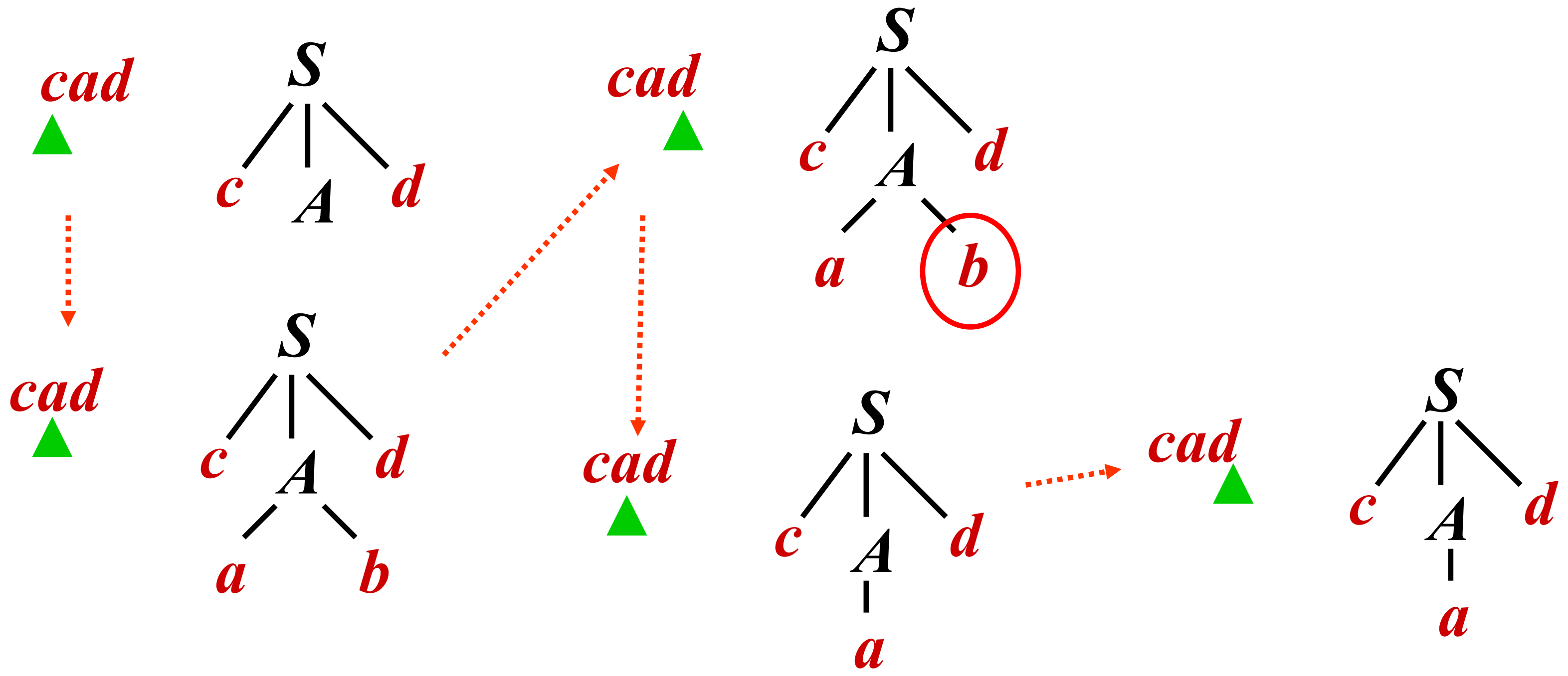
- **回到上一步, 重新选择A的第2个产生式**$A \rightarrow a$

例: 需要回溯的文法

考虑文法$S \rightarrow cAd$

$A \rightarrow ab \mid a$

为输入串$w = cad$ 建立分析树(假设按顺序选产生式)



- **成功匹配了输入串cad**

通用递归下降的问题: 回溯

- **复杂的回溯→代价太高**
  - 非终结符有可能有多个产生式，由于信息缺失，无法准确预测选择哪一个
  - 考虑到往往需要对多个非终结符进行推导展开，因此尝试的路径可能呈指数级爆炸

    例如：$A \rightarrow ab \mid a$，不知该选择哪条产生式

- **其分析过程类似于NFA**

是否可以构造一个类似于DFA的分析方法？

2. LL(1)和预测分析法

- LL(1)文法的定义
- 实现LL(1)预测分析
- 消除左递归、提左公因子
- 错误恢复

预测分析法(Predictive Parsing)

此方法接受LL(k)文法!

- L: “left-to-right” 从左到右扫描
- L: “leftmost derivation” 最左推导
- k: 向前看k个Token来确定产生式 (通常k=1)

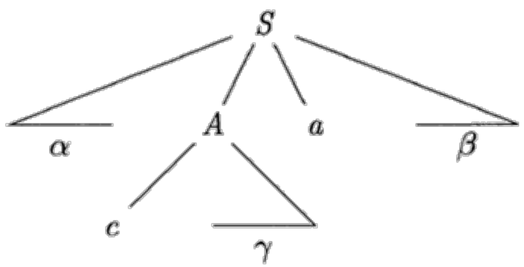
- 每次为最左边的非终结符号选择产生式时，向前看1个输入符号，预测要使用的产生式

文法加什么限制可以保证没有回溯？

First集和Follow集

给定 $G = (T, N, P, S)$, $\alpha \in (T \cup N)^*$

- $\text{First}(\alpha) = \{a \mid \alpha \Rightarrow^* a\ldots, a \in T\}$
  - 可从α推导得到的串的首个终结符的集合

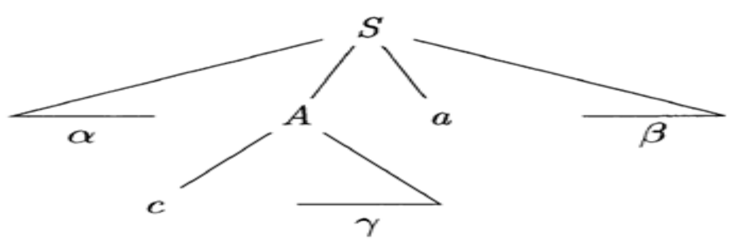


终结符号 c 在 FIRST(A)

First集和Follow集

给定 $G = (T, N, P, S)$, $A \in N$

- **Follow(A)** $= \{a \mid S \Rightarrow^* \ldots Aa\ldots, \ a \in T\}$
  - 从S出发, 可能在推导过程中跟在A右边的**终结符号**集
  - 例如:$S \rightarrow \alpha A a \beta$，终结符号a ∈ Follow($A$)



a 在 FOLLOW(A) 中

LL(1)文法的定义

- 文法G的任何两个产生式$A \rightarrow \alpha \mid \beta$ 都满足下列条件：

  1. $\mathrm{First}(\alpha) \cap \mathrm{First}(\beta) = \varnothing$

  α和β推导不出以同一个单词为首的串

  2. **若**$\beta \Rightarrow^* \varepsilon$ **，那么**$\alpha \not\Rightarrow^* \varepsilon$**，且**$\mathrm{First}(\alpha) \cap \mathrm{Follow}(A) = \varnothing$

  α和β不能同时推出ε; First(α)不应在Follow(A) 中

以上条件可以保证产生式选择的唯一性

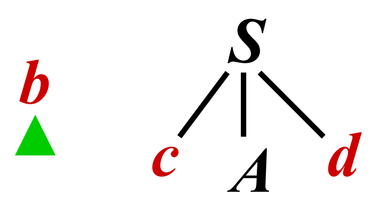
LL(1)文法的定义

- 文法G的任何两个产生式$A \rightarrow \alpha \mid \beta$ 都满足下列条件：

1. **First(α) ∩ First(β) = ∅**

α和β推导不出以同一个单词为首的串

意义: 假设下一个输入是b，且First(α)和First(β)不相交。若$b \in$ First(α)，则选择$A \rightarrow \alpha$；若$b \in$ First(β)，则选择$A \rightarrow \beta$



2. **若$\beta \Rightarrow^* \varepsilon$ ，那么$\alpha \not\Rightarrow^* \varepsilon$, 且First(α) ∩ Follow(A) = ∅**

α和β不能同时推出ε; First(α)不应在Follow(A) 中

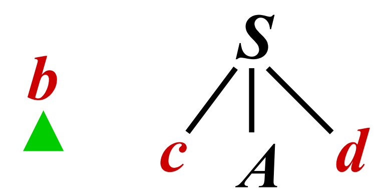
LL(1)文法的定义

- 文法G的任何两个产生式$A \rightarrow \alpha \mid \beta$ 都满足下列条件：

1. $\mathbf{First}(\alpha) \cap \mathbf{First}(\beta) = \varnothing$

α和β推导不出以同一个单词为首的串

意义: 假设下一个输入是b，且First(α)和First(β)不相交。若$b \in$ First(α)，则选择$A \rightarrow \alpha$；若$b \in$ First(β)，则选择$A \rightarrow \beta$



2. 若$\beta \Rightarrow^* \varepsilon$，那么$\alpha \not\Rightarrow^* \varepsilon$，且$\mathbf{First}(\alpha) \cap \mathbf{Follow}(A) = \varnothing$

α和β不能同时推出ε; First(α)不应在Follow(A) 中

意义: 假设下一个输入是b, 且 $\beta \Rightarrow^* \varepsilon$

- 如果$b \in$ First(α)，则选择$A \rightarrow \alpha$ *(属于上面条件1的情况)*
- 如果$b \in$ Follow(A)，则选择$A \rightarrow \beta$，因为A最终到达了ε且后面跟着b

2. LL(1)和预测分析法

- LL(1)文法的定义
- 实现LL(1)预测分析
  - 计算First, Follow→构造预测分析表→预测分析
- 消除左递归、提左公因子
- 错误恢复

回顾: First集和Follow集

给定 $G = (T, N, P, S)$, $\alpha \in (T \cup N)^*$, $A \in N$

- $\mathbf{First}(\alpha) = \{a \mid \alpha \Rightarrow^* a\ldots, a \in T\}$
  – 可从α推导得到的串的首个终结符的集合

- $\mathbf{Follow}(A) = \{a \mid S \Rightarrow^* \ldots Aa\ldots, a \in T\}$
  - 从S出发, 可能在推导过程中跟在A右边的**终结符号**集

Nullable集的归纳定义

- First, Follow集涉及**空串**，我们引入Nullable概念

- If X can derive an empty string (**X is Nullable**), **iff**
  - **Base case**:
    - $X \rightarrow \varepsilon$: then X must be Nullable
  - **Inductive case**:
    - $X \rightarrow Y_1 \ldots Y_n$

      If $Y_1, \ldots, Y_n$ are n nonterminals and **may all derive** empty strings, then X is Nullable.

根据归纳定义计算Nullable集

- Compute the **Nullable set** by iteration:

```
/* Nullable: a set of nonterminals */
Nullable <- {};
while (Nullable still changes)
  for (each production X → α)
    switch (α)
    case ε:
      Nullable U= {X};
      break;
    case Y1 … Yn:
      if (Y1∈Nullable && … && Yn∈Nullable)
        Nullable U= {X};
      break;
```

First集的归纳定义

$$G = (T, N, P, S),\ \alpha \in (T \cup N)^*,\ A \in N$$

- $\mathrm{First}(\alpha) = \{a \mid \alpha \Rightarrow^* a\ldots,\ a \in T\}$
  可从α推导得到的串的首终结符号集合
  - **Base case:**
    - If X is a terminal: First (X) = {X}
  - **Inductive case:**
    - If $X \rightarrow Y_1\ Y_2 \ldots Y_n$
      - First (X) ∪= First(Y_1)
      - If $Y_1 \in$Nullable, First (X) ∪= First(Y_2)
      - If Y_1,Y_2 $\in$Nullable, First (X) ∪= First(Y_3)
      - ……

注: 上述规则似乎是关于非终结符的。但是First是关于文法符号串α（如产生式右部) 的，计算规则如inductive case

Follow集的归纳定义

$$G = (T, N, P, S),\ \alpha \in (T \cup N)^*,\ A \in N$$

- **Follow(A)** = $\{a \mid S \Rightarrow^* \ldots Aa\ldots,\ a \in T\}$
  从S出发, 可能在推导过程中跟在A右边的终结符号集
  - **Base case:**
    - Follow (A) = {}
  - **Inductive case:**
    - 假设存在产生式 B → s1 A s2 for any s1 and s2
      1. Follow(A) ∪= First(s2)
      2. If s2 is Nullable, Follow(A) ∪= Follow(B)

- 关于第2种情况, 假设S =>* …B b… (b属于Follow(B))
  - 用s1 A s2替换B后：S =>* …s1 A s2 b …
  - 由于s2 is Nullable, 因此b也属于Follow(A)!

例: 计算Nullable, First和Follow集

$Z \to d$	$Y \to c$	$X \to Y$
$Z \to X\ Y\ Z$	$Y \to \varepsilon$	$X \to a$

注意：有的地方(如虎书) 写作

$Y \to c$

$Y \to$

第2个产生式”$Y \to$ “也表示” $Y \to \varepsilon$”

例: 计算Nullable, First, Follow集

$Z \rightarrow d$
$Z \rightarrow X\ Y\ Z$

$Y \rightarrow c$
$Y \rightarrow \varepsilon$

$X \rightarrow Y$
$X \rightarrow a$

- 初始化为False

	nullable	first	follow
Z	False		
Y	False		
X	False		

例: 计算Nullable, First, Follow集

$Z \rightarrow d$	$Y \rightarrow c$	$X \rightarrow Y$
$Z \rightarrow X\ Y\ Z$	$\boxed{Y \rightarrow \varepsilon}$	$X \rightarrow a$

- Since $Y \rightarrow \varepsilon$, Nullable ∪= {Y}; (Base Case)

	nullable	first	follow
Z	False		
Y	True		
X	False		

例: 计算Nullable, First, Follow集

$Z \to d$
$Z \to X\ Y\ Z$

$Y \to c$
$Y \to \varepsilon$

$\boxed{X \to Y}$
$X \to a$

- Since $Y \to \varepsilon$， Nullable ∪= {Y}; (Base Case)
- Since $X \to Y$ and Y is Nullalbe， Nullable ∪= {X} (Inductive Case)

	nullable	first	follow
Z	False		
Y	True		
X	True		

例: 计算Nullable, First, Follow集

$Z \to d$

$\boxed{Z \to X\ Y\ Z}$

$Y \to c$

$Y \to \varepsilon$

$X \to Y$

$X \to a$

Stop iteration

	nullable	first	follow
Z	False		
Y	True		
X	True		

例: 计算Nullable, First, Follow集

$Z \rightarrow d$
$Z \rightarrow X\ Y\ Z$

$Y \rightarrow c$
$Y \rightarrow \varepsilon$

$X \rightarrow Y$
$X \rightarrow a$

Nullable = {X, Y}

	nullable	first	follow
Z	False	{}	
Y	True	{}	
X	True	{}	

例: 计算Nullable, First, Follow集

$Z \to d$
$Z \to X\ Y\ Z$

$Y \to c$
$Y \to \varepsilon$

$X \to Y$
$X \to a$

Nullable = {X, Y}

	nullable	first	follow
Z	False	{d}	
Y	True	{c}	
X	True	{a}	

If X is a terminal: First (X) = {X}
If X -> $Y_1\ Y_2\ \ldots\ Y_n$

- First (X) ∪= First(Y_1)
- If $Y_1 \in$Nullable, First (X) ∪= First(Y_2)
- If Y_1,Y_2 $\in$Nullable, First (X) ∪= First(Y_3)
- ……

注: 有时候可利用归纳定义的简单推论：
If X -> a (a $\in$ *T*): First (X) ∪= {a}

例: 计算Nullable, First, Follow集

$Z \to d$
$Z \to X\ Y\ Z$

$Y \to c$
$Y \to \varepsilon$

$\boxed{X \to Y}$
$X \to a$

Nullable = {X, Y}

	nullable	first	follow
Z	False	{d}	
Y	True	{c}	
X	True	{a, c}	

If X is a terminal: First (X) = {X}

If X -> $Y_1\ Y_2 \ldots Y_n$

- First (X) ∪= First(Y_1)
- If $Y_1 \in$Nullable, First (X) ∪= First(Y_2)
- If $Y_1, Y_2 \in$Nullable, First (X) ∪= First(Y_3)
- ……

例: 计算Nullable, First, Follow集

$Z \to d$
$\boxed{Z \to X\ Y\ Z}$

$Y \to c$
$Y \to \varepsilon$

$X \to Y$
$X \to a$

Nullable = {X, Y}

	nullable	first	follow
Z	False	{d, a, c}	
Y	True	{c}	
X	True	{a, c}	

If X is a terminal: First (X) = {X}

If X -> $Y_1\ Y_2 \ldots Y_n$

- First (X) ∪= First(Y_1)
- If $Y_1 \in$ Nullable, First (X) ∪= First(Y_2)
- If $Y_1, Y_2 \in$ Nullable, First (X) ∪= First(Y_3)
- ……

例: 计算Nullable, First, Follow集

$Z \rightarrow d$
$Z \rightarrow X\ Y\ Z$

$Y \rightarrow c$
$Y \rightarrow \varepsilon$

$X \rightarrow Y$
$X \rightarrow a$

Nullable = {X, Y}

	nullable	first	follow
Z	False	{d, a, c}	
Y	True	{c}	
X	True	{a, c}	

Iterate again.
No change.
Stop iteration.

例: 计算Nullable, First, Follow集

$Z \rightarrow d$
$Z \rightarrow X\ Y\ Z$

$Y \rightarrow c$
$Y \rightarrow \varepsilon$

$X \rightarrow Y$
$X \rightarrow a$

Nullable = {X, Y}

Initialize follow sets

	nullable	first	follow
Z	False	{d, a, c}	{}
Y	True	{c}	{}
X	True	{a, c}	{}

例: 计算Nullable, First, Follow集

Z → d
Z → X $\boxed{Y}$ Z

Y → c
Y → ε

X → Y
X → a

Nullable = {X, Y}

利用Inductive Case的情况 1: Follow(Y) U= First(Z) = ?

	nullable	first	follow
Z	False	{d, a, c}	{}
Y	True	{c}	{d, a, c}
X	True	{a, c}	{}

B → s1 A s2 for any s1, s2

1. Follow(A) ∪= First(s2)
2. If s2 is Nullable:
   Follow(A) ∪= Follow(B)

例: 计算Nullable, First, Follow集

$Z \to d$
$Z \to \boxed{X}\, Y\, Z$

$Y \to c$
$Y \to \varepsilon$

$X \to Y$
$X \to a$

Nullable = {X, Y}

利用Inductive Case的情况 1: Follow(X) ∪= First(YZ) = ?

	nullable	first	follow
Z	False	{d, a, c}	{}
Y	True	{c}	{d, a, c}
X	True	{a, c}	{c, d, a}

$B \to$ s1 A s2 for any s1, s2

1. Follow(A) ∪= First(s2)
2. If s2 is Nullable:
   Follow(A) ∪= Follow(B)

例: 计算Nullable, First, Follow集

$Z \to d$

$\boxed{Z \to X\ Y\ Z}$

$\boxed{Y \to c}$

$\boxed{Y \to \varepsilon}$

$X \to Y$

$\boxed{X \to a}$

Nullable = {X, Y}

	nullable	first	follow
Z	False	{d, a, c}	{}
Y	True	{c}	{d, a, c}
X	True	{a, c}	{c, d, a}

$B \to s1\ A\ s2$ for any s1, s2

1. Follow(A) ∪= First(s2)
2. If s2 is Nullable:
   Follow(A) ∪= Follow(B)

例: 计算Nullable, First, Follow集

$Z \rightarrow d$
$Z \rightarrow X\ Y\ Z$

$Y \rightarrow c$
$Y \rightarrow \varepsilon$

$\boxed{X \rightarrow Y}$
$X \rightarrow a$

Nullable = {X, Y}

Follow(Y) ∪= Follow(X)

	nullable	first	follow
Z	False	{d, a, c}	{}
Y	True	{c}	{d, a, c}
X	True	{a, c}	{c, d, a}

$B \rightarrow$ s1 A s2 for any s1, s2

1. Follow(A) ∪= First(s2)
2. If s2 is Nullable:
   Follow(A) ∪= Follow(B)

例: 计算Nullable, First, Follow集

$Z \to d$
$Z \to X\ Y\ Z$

$Y \to c$
$Y \to \varepsilon$

$X \to Y$
$X \to a$

Nullable = {X, Y}

	nullable	first	follow
Z	False	{d, a, c}	{}
Y	True	{c}	{d, a, c}
X	True	{a, c}	{c, d, a}

优化Nullable, First, Follow的计算

- 上述过程按顺序算了Nullable, First, Follow集
- 实际上，它们可以同时计算!
  - See Tiger book algorithm 3.13

Initialize FIRST and FOLLOW to all empty sets, and nullable to all false.
for each terminal symbol Z
    FIRST[Z] $\leftarrow$ {Z}
repeat
    for each production $X \rightarrow Y_1 Y_2 \cdots Y_k$
      for each i from 1 to k, each j from $i+1$ to k,
        if all the Y_i are nullable
          then nullable[X] $\leftarrow$ true
        if $Y_1 \cdots Y_{i-1}$ are all nullable
          then FIRST[X] $\leftarrow$ FIRST[X] $\cup$ FIRST[Y_i]
        if $Y_{i+1} \cdots Y_k$ are all nullable
          then FOLLOW[Y_i] $\leftarrow$ FOLLOW[Y_i] $\cup$ FOLLOW[X]
        if $Y_{i+1} \cdots Y_{j-1}$ are all nullable
          then FOLLOW[Y_i] $\leftarrow$ FOLLOW[Y_i] $\cup$ FIRST[Y_j]
until FIRST, FOLLOW, and nullable did not change in this iteration.

不要求必需掌握，选适合的即可😄(有人可能觉得Alg. 3.13更好用)

2. LL(1)和预测分析法

- LL(1)文法的定义
- 实现LL(1)预测分析
  - 计算First, Follow→构造预测分析表→预测分析
- 消除左递归、提左公因子
- 错误恢复

回顾: 自顶向下语法分析

- 从分析树的顶部向底部方向构造分析树
  - 从文法开始符号S推导出串w

- 每一步推导中，都需要做两个选择
  1. 替换当前句型中的哪个非终结符?
     - 总是选择每个句型的最左非终结符进行替换!

  2. **用该非终结符的哪个产生式进行替换?**

预测分析表

- 表驱动分析程序需要的二维表M
  - 表的每一行A对应一个非终结符
  - 表的每一列a 对应某个终结符或输入结束符$
  - 表中的项M(A,a) 表示: 针对**非终结符为A，当下一个输入Token为a时，可选的产生式集合**

	int	*	+	$
T	int Y			
E	T X			
X			+ E	ε
Y		* T	ε	ε

Consider the [E, int] entry:

"When current non-terminal is E and next input is int, use production $E \rightarrow T\ X$

预测分析表的构造

Grammar:

$Z \to X\ Y\ Z$　　$Y \to c$　　$X \to a$

$Z \to d$　　$Y \to \varepsilon$　　$X \to Y$

	nullable	first	follow
Z	no	d,a,c	
Y	yes	c	a,c,d
X	yes	a,c	a,c,d

对文法G的每个产生式$X \to \gamma$

- **if** t $\in$ First(γ): enter ($X \to \gamma$) in row X, col t
- **if** γ is Nullable and t $\in$ Follow(X): enter ($X \to \gamma$) in row X, col t

	a	c	d
Z			
Y			
X			

预测分析表的构造

Grammar:

$\boxed{Z \rightarrow X\,Y\,Z}$ $\quad Y \rightarrow c \quad X \rightarrow a$

$Z \rightarrow d \quad Y \rightarrow \varepsilon \quad X \rightarrow Y$

	nullable	first	follow
Z	no	d,a,c	
Y	yes	c	a,c,d
X	yes	a,c	a,c,d

对文法G的每个产生式$X \rightarrow \gamma$

- if t $\in$ First(γ): enter ($X \rightarrow \gamma$) in row X, col t
- if γ is Nullable and t $\in$ Follow(X): enter ($X \rightarrow \gamma$) in row X, col t

	a	c	d
Z	Z→XYZ	Z→XYZ	Z→XYZ
Y			
X			

First(XYZ)

预测分析表的构造

Grammar:

$Z \to X\,Y\,Z$　　$\boxed{Y \to c}$　　$\boxed{X \to a}$

$\boxed{Z \to d}$　　$Y \to \varepsilon$　　$X \to Y$

	nullable	first	follow
Z	no	d,a,c	
Y	yes	c	a,c,d
X	yes	a,c	a,c,d

对文法G的每个产生式$X \to \gamma$

- **if** $t \in$ First(γ): enter ($X \to \gamma$) in row X, col t
- **if** γ is Nullable and $t \in$ Follow(X): enter ($X \to \gamma$) in row X, col t

	a	c	d
Z	Z→XYZ	Z→XYZ	Z→XYZ Z→d
Y		Y→c	
X	X→a		

预测分析表的构造

Grammar:

$Z \rightarrow X\,Y\,Z$ $\quad Y \rightarrow c$ $\quad X \rightarrow a$

$Z \rightarrow d$ $\quad \boxed{Y \rightarrow \varepsilon}$ $\quad X \rightarrow Y$

	nullable	first	follow
Z	no	d,a,c	
Y	yes	c	a,c,d
X	yes	a,c	a,c,d

对文法G的每个产生式$X \rightarrow \gamma$

- if t $\in$ First(γ): enter ($X \rightarrow \gamma$) in row X, col t
- if γ is Nullable and t $\in$ Follow(X): enter ($X \rightarrow \gamma$) in row X, col t

	a	c	d
Z	Z→XYZ	Z→XYZ	Z→XYZ Z→d
Y	Y→ε	Y→c Y→ε	Y→ε
X	X→a		

Follow(Y) = {a, c, d}

预测分析表的构造

Grammar:

$Z \to X\,Y\,Z$　　$Y \to c$　　$X \to a$

$Z \to d$　　$Y \to \varepsilon$　　$\boxed{X \to Y}$

	nullable	first	follow
Z	no	d,a,c	
Y	yes	c	a,c,d
X	yes	a,c	a,c,d

对文法G的每个产生式$X \to \gamma$

- **if t $\in$ First(γ): enter ($X \to \gamma$) in row X, col t**
- **if γ is Nullable and t $\in$ Follow(X): enter ($X \to \gamma$) in row X, col t**

	a	c	d
Z	Z→XYZ	Z→XYZ	Z→XYZ Z→d
Y	Y→ε	Y→c Y→ε	Y→ε
X	X→a X→Y	X→Y	X→Y

Follow(X) = {a, c, d}

预测分析表的构造

Grammar:

$Z \rightarrow X\,Y\,Z$ $\quad Y \rightarrow c \quad X \rightarrow a$

$Z \rightarrow d$ $\quad Y \rightarrow \varepsilon \quad X \rightarrow Y$

	nullable	first	follow
Z	no	d,a,c	
Y	yes	c	a,c,d
X	yes	a,c	a,c,d

对文法G的每个产生式$X \rightarrow \gamma$

- if t $\in$ First(γ): enter ($X \rightarrow \gamma$) in row X, col t
- if γ is Nullable and t $\in$ Follow(X): enter ($X \rightarrow \gamma$) in row X, col t

	a	c	d
Z	Z→XYZ	Z→XYZ	Z→XYZ Z→d
Y	Y→ε	Y→c Y→ε	Y→ε
X	X→a X→Y	X→Y	X→Y

预测分析表的构造

Grammar:

$Z \to X\ Y\ Z$ $\quad Y \to c \quad X \to a$

$Z \to d \quad Y \to \varepsilon \quad X \to Y$

	nullable	first	follow
Z	no	d,a,c	
Y	yes	c	a,c,d
X	yes	a,c	a,c,d

What are the blanks? --> syntax errors

	a	c	d
Z	**Z→XYZ**	**Z→XYZ**	**Z→XYZ** **Z→d**
Y	**Y→ε**	**Y→c** **Y→ε**	**Y→ε**
X	**X→a** **X→Y**	**X→Y**	**X→Y**

预测分析表的构造

Grammar:

Z → X Y Z Y → c X → a
Z → d Y → ε X → Y

	nullable	first	follow
Z	no	d,a,c	
Y	yes	c	a,c,d
X	yes	a,c	a,c,d

Is it possible to put 2 grammar rules in the same box?

	a	c	d
Z	Z→XYZ	Z→XYZ	Z→XYZ Z→d
Y	Y→ε	Y→c Y→ε	Y→ε
X	X→a X→Y	X→Y	X→Y

上面的文法不是LL(1)文法！！

利用预测分析表定义LL(1)文法

- 文法G的任何两个产生式$A \rightarrow \alpha \mid \beta$ 都满足下列条件：

  1. **First(α) ∩ First(β) = ∅**

  α和β推导不出以同一个单词为首的串

  2. **若β ⇒* ε ，那么α ⇏* ε , 且First(α) ∩ Follow(A) = ∅**

  α和β不能同时推出ε; First(α)不应在Follow(A) 中

- **LL(1) 文法**:

  If a predictive parsing table constructed this way contains no duplicate entries, the grammar is called LL(1)!

  Left-to-right parse, left-most derivation, 1 symbol lookahead

2. LL(1)和预测分析法

- LL(1)文法的定义
- **实现LL(1)预测分析**
  - 计算First, Follow→构造预测分析表→**预测分析**
- 消除左递归、提左公因子
- 错误恢复

LL(1)分析的实现

- **递归下降LL(1)分析**
  - 递归下降分析: 非终结符对应子过程

- **非递归LL(1)分析**
  - 使用显式的栈，而不是递归调用来完成分析(类似模拟下推自动机PDA)

LL(1) 的递归下降实现

- 递归下降语法分析程序由一组过程组成
- 每个非终结符号对应于一个过程
- **可以通过向前看一个输入符号来唯一地选择产生式**

```
void A( ) {
    选择一个A产生式，A→X1 X2 ... Xk ;
    for ( i = 1 to k ) {
        if ( Xi是一个非终结符号)
            调用过程 Xi( ) ;
        else if ( Xi 等于输入符号a)
            读入下一个输入符号;
        else /* 发生了一个错误 */ ;
            ? ? ?
    }
}
```

如果Lookahead输入符号为 b，那么选择 $M[A, b]$ 中的产生式

例: LL(1) 的递归下降实现

S -> if E then S else S
S -> begin S L
S -> print E

L -> end
L -> ; S L
E -> num = num

- Step 1: Represent the token

```
enum token {IF, THEN, ELSE, BEGIN, END, PRINT, SEMI,
NUM, EQ};
```

例: LL(1) 的递归下降实现

S -> if E then S else S
S -> begin S L
S -> print E

L -> end
L -> ; S L
E -> num = num

- Step 2: build infrastructure for reading tokens from lexer

```
// call lexer
extern enum token getToken(void);


// store the next token
enum token tok;
void advance() {tok=getToken();}
// consume the next token and get the new one
void eat(enum token t) {if (tok==t) advance(); else
error();}
```

例: LL(1) 的递归下降实现

S -> if E then S else S
S -> begin S L
S -> print E

L -> end
L -> ; S L
E -> num = num

- Step 3: build a function for each non-terminal

```
void S(void) {
  switch(tok) {
    case IF: eat(IF); E(); eat(THEN); S(); eat(ELSE); S(); break;
    case BEGIN: eat(BEGIN); S(); L(); break;
    case PRINT: eat(PRINT); E(); break;
    default: error(); }
}
void L(void) {
  switch(tok) {
    case END: eat(END); break;
    case SEMI: eat(SEMI); S(); L(); break;
    default: error(); }
}
void E(void) { eat(NUM); eat(EQ); eat(NUM); }
```

例: LL(1) 的递归下降实现

- This grammar is very special
  - For each non-terminal, the first symbols Y1 in the right-hand sides of its productions are different terminals

```
S -> if E then S else S        L -> end
S -> begin S L                 L -> ; S L
S -> print E                   E -> num = num
```

例: LL(1) 的递归下降实现

考虑下面的新文法:

S -> E$
E -> E + T
E -> E – T
E -> T

$T \rightarrow T * F$
$T \rightarrow T / F$
$T \rightarrow F$

F -> id
F -> num
F -> (E)

- Step 3: build a function for each non-terminal

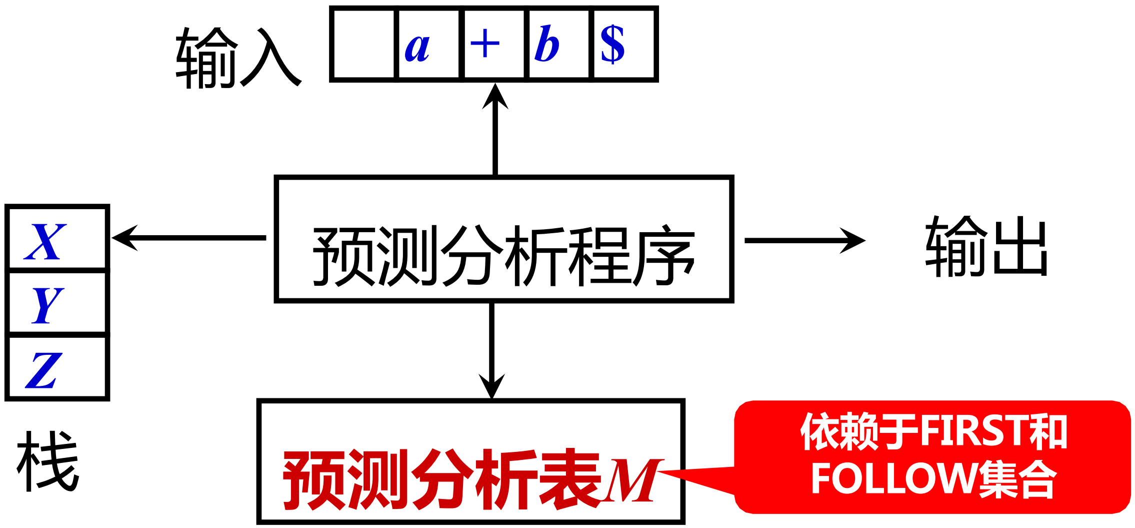
```
void S(void) { E(); eat(EOF); }
void E(void) {
   switch (tok) {
     case ?: E(); eat(PLUS); T(); break;
     case ?: E(); eat(MINUS); T(); break;
     case ?: T(); break;
     default: error();
   }
}
```

If tok == num, which production to choose?

此时需要预测分析表(在实现时可能是“硬编码”到switch语句)

LL(1)的非递归实现(不要求掌握)

- 可以看作是实现LL(1)对应的的PDA(pushdown automata)
- 针对输入w的两个基本动作
  - 如果栈顶是非终结符A：利用预测分析表, 选择产生式$A \rightarrow \alpha$ (也就是将栈顶的非终结符A替换成串α)
  - 如果栈顶是终结符a：将栈顶记号a和输入中的Token匹配



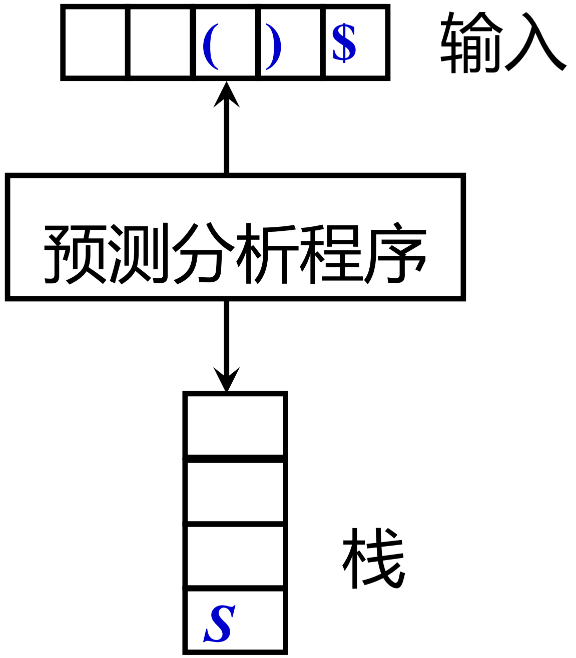
初始状态：$w\$$在输入缓冲区中。

例: 非递归分析 配对括号文法

对串（ ），下表给出自顶向下的分析程序的动作：

- 0. 第"0"步: 先把文法开始符号S压栈

步骤	分析栈	输入	动作
1	S	( ) \$	$S \to (S)\, S$
2	S) S (	( ) \$	匹配
3	S) S	) \$	$S \to \varepsilon$
4	S)	) \$	匹配
5	S	\$	$S \to \varepsilon$
6		\$	接受

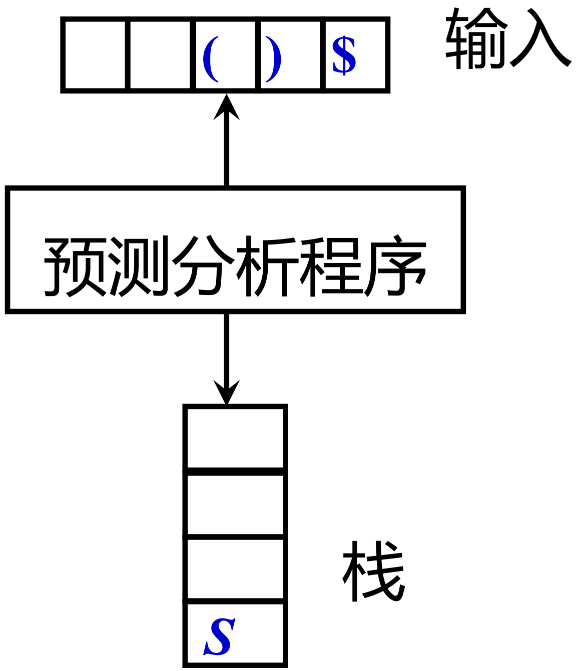


例: 非递归分析 配对括号文法

对串（ ），下表给出自顶向下的分析程序的动作：

1. **动作:** $S \rightarrow (S)\ S$，根据产生式对栈顶S进行替换
   - 先Pop栈顶S
   - 再挨个Push S), S, (

步骤	分析栈	输入	动作
1	S	( ) \$	$S \rightarrow (S)\ S$
2	S) S (	( ) \$	匹配
3	S) S	) \$	$S \rightarrow \varepsilon$
4	S)	) \$	匹配
5	S	\$	$S \rightarrow \varepsilon$
6		\$	接受



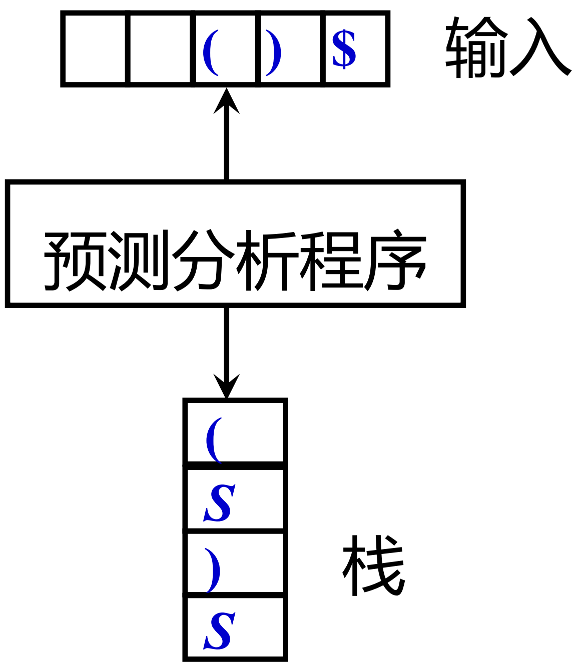
例: 非递归分析 配对括号文法

对串（ ），下表给出自顶向下的分析程序的动作：

2. 动作：匹配，栈顶元素 “(“ 和输入符号 “(“ 匹配成功！

- Pop栈顶元素
- 输入往右移

步骤	分析栈	输入	动作
1	S	( ) \$	$S \to (S)\ S$
2	$S\ )\ S\ ($	( ) \$	匹配
3	$S\ )\ S$	) \$	$S \to \varepsilon$
4	$S\ )$	) \$	匹配
5	S	\$	$S \to \varepsilon$
6		\$	接受

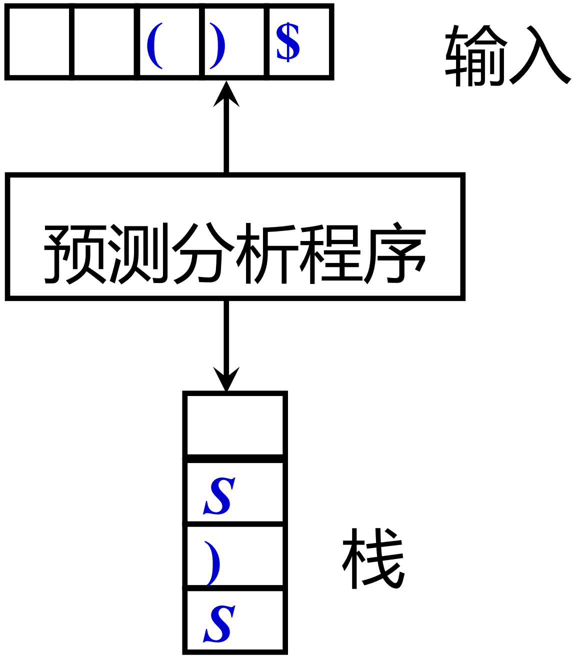


例: 非递归分析 配对括号文法

对串（ ），下表给出自顶向下的分析程序的动作：

3. 动作：$S \rightarrow \varepsilon$，根据产生式对栈顶S进行替换
   1. Pop栈顶元素S
   2. Push空串ε (相当于只把S给pop了)

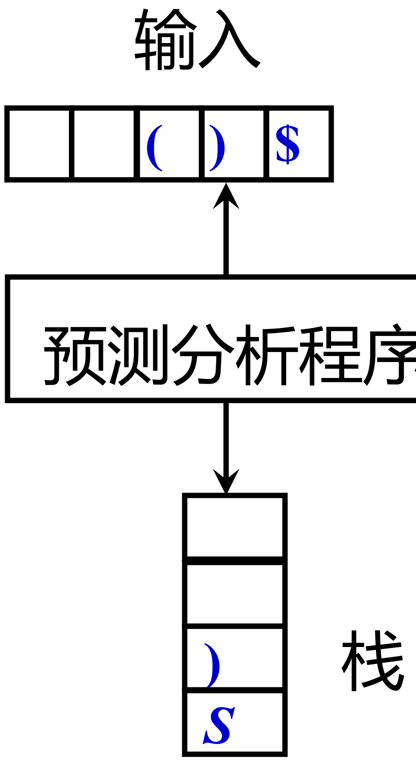
步骤	分析栈	输入	动作
1	S	( ) \$	$S \rightarrow (S)\ S$
2	S) S (	( ) \$	匹配
3	S) S	) \$	$S \rightarrow \varepsilon$
4	S)	) \$	匹配
5	S	\$	$S \rightarrow \varepsilon$
6		\$	接受



例: 非递归分析 配对括号文法

对串（ ），下表给出自顶向下的分析程序的动作：

步骤	分析栈	输入	动作
1	S	( ) \$	$S \to (S)\ S$
2	S) S (	( ) \$	匹配
3	S) S	) \$	$S \to \varepsilon$
4	S)	) \$	匹配
5	S	\$	$S \to \varepsilon$
6		\$	接受



剩下的步骤略去...

2. 预测分析法和LL(1)

- □ LL(1)文法的定义
- □ 实现LL(1)预测分析
- □ 消除左递归、提左公因子
- □ 错误恢复

LL(1)文法的重要性质

- **LL(1)文法有一些明显的性质**
  - LL(1)文法是无二义的
  - LL(1)文法是无左递归的
  - LL(1)文法是无左公因子的

除了利用定义外，有时可以利用这些性质判定某些文法不是LL(1)的

LL(1)文法要求: 无左递归

- **左递归(left-recursive)文法**
  - 如果一个文法中有非终结符号A使得$A \Rightarrow^{+} A\alpha$，那么这个文法就是左递归的
  - $S \rightarrow Sa \mid b$ **(直接/立即左递归)**

- **问题: 递归下降分析可能进入无限循环**
  - 如考虑串**baaaaa**
  - 最左推导: $S \Rightarrow Sa \Rightarrow Saa \Rightarrow Saaa \Rightarrow Saaaa$ ....

思路: 限制文法或进行文法变换

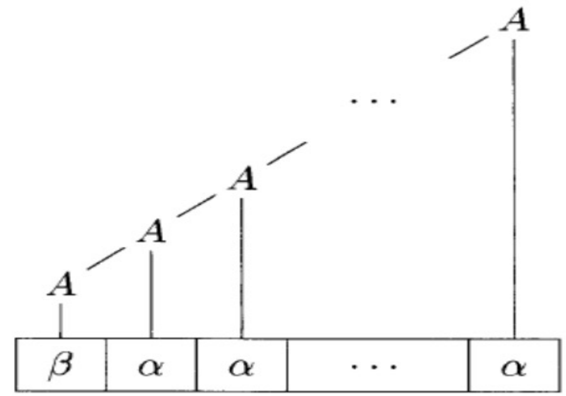
文法变换: 消除直接左递归

$A \to A\alpha \mid \beta$ ，其中$\alpha \neq \varepsilon$,α，β 不以A开头

$$A \to \beta A'$$

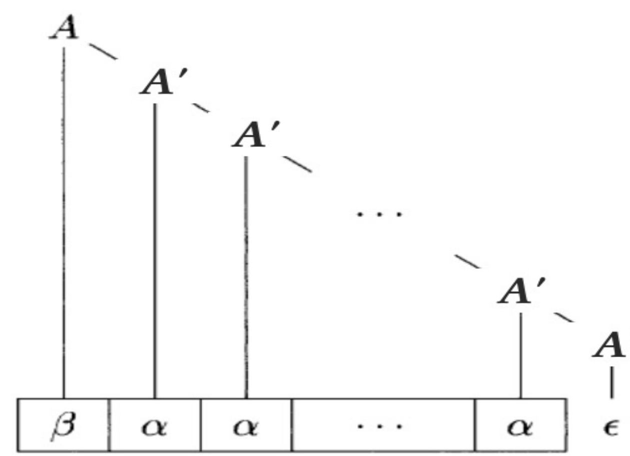
$$A' \to \alpha A' \mid \varepsilon$$

把左递归转成了右递归



a)

- 观察：由A生成的串以某个β开头，然后跟上零个或多个α



b)

通用的左递归消除方法(详见龙书)

LL(1)文法要求：无左公因子

- 有左公因子的(left-factored)文法
  - $P \to \alpha\beta \mid \alpha\gamma$

- 问题: 同一非终结符的多个候选式存在共同前缀，可能导致回溯

思路: 限制文法或进行文法变换

文法变换: 提左公因子

- 有**左公因子**的(left-factored)文法
  - $P \rightarrow \alpha\beta \mid \alpha\gamma$
- 提左公因子(left factoring)
  - 对形如 $P \rightarrow \alpha\beta \mid \alpha\gamma$ 的一对产生式，用如下产生式替换

    $P \rightarrow \alpha Q$

    $Q \rightarrow \beta \mid \gamma$

  其中Q为新增加的未出现过的非终结符

通过改写产生式来推迟决定，等读入了足够多的输入，获得足够信息后再做选择

2. 预测分析法和LL(1)

- LL(1)文法的定义
- 实现LL(1)预测分析
- 消除左递归、提左公因子
- 错误恢复

Predictive Parsing – Error Recovery

- A blank entry indicates a syntax error
- How should errors be handled?
  - Raise an exception and quit parsing
  - print an error message and recover from the error
- Raise an exception and quit:

```
void T( ) {
  switch (tok) {
    case ID:
    case NUM:
    case LPAREN: F(); Tprime(); break;
    default: error!
  }
}
```

Predictive Parsing – Error Recovery

- print an error message and recover from the error
- Errors can be recovered by deleting, replacing, or inserting tokens.
- Through inserting: pretend we have the token and return normally

```
void T( ) {
  switch (tok) {
    case ID:
    case NUM:
    case LPAREN: F( ); Tprime( ); break;
    default: print("expected id, num, or left-
paren");
  }}
```

may not terminate

- Deleting tokens is safer, because the loop must eventually terminate when EOF is reached

Predictive Parsing – Error Recovery

- Simple recover by deletion works by skipping tokens util a token in the FOLLOW set is reached.

```
int Tprime_follow [ ] = {PLUS, RPAREN, EOF};
void Tprime( ) {
  switch (tok) {
    case PLUS: break;
    case TIMES: eat(TIMES); F(); Tprime(); break;
    case RPAREN: break;
    case EOF: break;
    default: print("expected +, *, right-paren, or
end-of-file");
    skipto(Tprime_follow);
  }
}
```

本讲小结

- **Recursive descent parser for LL(k) grammars by:**
  - Computing nullable, first and follow sets
  - Constructing a parse table from the sets
  - Checking for duplicate entries, which indicates failure
  - Creating an C program from the parse table

- **If parser construction fails, we can**
  - Rewrite the grammar (left factoring, eliminating left recursion, etc.) and try again
  - Try to build a parser using some other method

Thank you all for your attention

Nullable和First的另一种计算方法

- Follow Set那个归纳定义 B -> s1 A s2是”一般“情况，对于一些比较特殊的产生式，可能可以用简单的”推论“，比如

Production	Constraints
$T' \rightarrow T\$$	$\{\$\} \subseteq FOLLOW(T)$
$T \rightarrow R$	$FOLLOW(T) \subseteq FOLLOW(R)$
$T \rightarrow \mathtt{a}T\mathtt{c}$	$\{\mathtt{c}\} \subseteq FOLLOW(T)$
$R \rightarrow \varepsilon$	
$R \rightarrow R\mathtt{b}R$	$\{\mathtt{b}\} \subseteq FOLLOW(R)$

Nullable和First的另一种计算方法

$$
\begin{aligned}
\mathit{Nullable}(\varepsilon) &= \mathit{true} \\
\mathit{Nullable}(\mathtt{a}) &= \mathit{false} \\
\mathit{Nullable}(\alpha\beta) &= \mathit{Nullable}(\alpha) \wedge \mathit{Nullable}(\beta) \\
\mathit{Nullable}(N) &= \mathit{Nullable}(\alpha_1) \vee \ldots \vee \mathit{Nullable}(\alpha_n), \\
&\quad \textit{where the productions for } N \textit{ are} \\
&\quad N \to \alpha_1, \quad \ldots \quad , N \to \alpha_n
\end{aligned}
$$

where $\mathtt{a}$ *is a terminal,* N *is a nonterminal,* α *and* β *are sequences of grammar symbols and* ε *represents the empty sequence of grammar symbols.*

$$
\begin{aligned}
\mathit{FIRST}(\varepsilon) &= \emptyset \\
\mathit{FIRST}(\mathtt{a}) &= \{\mathtt{a}\} \\
\mathit{FIRST}(\alpha\beta) &= \begin{cases} \mathit{FIRST}(\alpha) \cup \mathit{FIRST}(\beta) & \textit{if } \mathit{Nullable}(\alpha) \\ \mathit{FIRST}(\alpha) & \textit{if not } \mathit{Nullable}(\alpha) \end{cases} \\
\mathit{FIRST}(N) &= \mathit{FIRST}(\alpha_1) \cup \ldots \cup \mathit{FIRST}(\alpha_n) \\
&\quad \textit{where the productions for } N \textit{ are} \\
&\quad N \to \alpha_1, \quad \ldots \quad , N \to \alpha_n
\end{aligned}
$$

LL(1)文法的非递归预测分析(不要求掌握)

- 输入:串 w，预测分析表 M
- 输出:如果 w 是句子，输出 w 的最左推导;否则报错

置 ip 指向 $w\$$ 的第一个符号，栈为\$和开始符号S
repeat
令X是栈顶符号，a是ip所指向的符号；
if (X 是一个终结符号或\$){
if (X=a) 从栈中弹出X，ip指向下一个符号
else *error*();
} **else** { /*X是非终结符号*/
if ($M[X,a]=X\rightarrow Y_1Y_2\ldots\ldots Y_k$) {
从栈中弹出X;
将 $Y_k, Y_{k-1}, \ldots\ldots, Y_1$压入栈中，即$Y_1$在栈顶；
}
else *error*();
}
until X=\$ /*栈为空*/

Input pointer
就是lookahead

例: 预测分析器在id+id*id上的动作？

- **文法** $E \to TE'$
  $E' \to + TE' \mid \varepsilon$
  $T \to FT'$
  $T' \to * FT' \mid \varepsilon$
  $F \to (E) \mid \mathrm{id}$

FIRST(E) = { (, id }
FIRST(E') = {+, ε}
FRIST(T') = {*, ε}
FOLLOW(E) = {), $}
FOLLOW(T) = FOLLOW (T') = {+,), $}
FOLLOW(F) = {+, *,), $}

非终结符	输入符号					
	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to \mathrm{id}$			$F \to (E)$		

- **行：非终结符；列：终结符或$ ；单元：产生式**

例: 预测分析器在id+id*id上的动作(一)

STACK	INPUT	OUTPUT
$\$E$	id + id * id\$	
$\$E'T$	id + id * id\$	$E \rightarrow TE'$
$\$E'T'F$	id + id * id\$	$T \rightarrow FT'$
$\$E'T'$id	id + id * id\$	$F \rightarrow$ id
$\$E'T'$	+ id * id\$	
$\$E'$	+ id * id\$	$T' \rightarrow \varepsilon$
$\$E'T+$	+ id * id\$	$E' \rightarrow +TE'$
$\$E'T$	id * id\$	

	输入符号					
	id	+	*	(	)	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow$ id			$F \rightarrow (E)$		

例：预测分析器在id+id*id上的动作(二)

STACK	INPUT	OUTPUT
$\$E'T'F$	id * id$	$T \to FT'$
$\$E'T'$**id**	id * id$	$F \to$ **id**
$\$E'T'$	* id$	
$\$E'T'F*$	* id$	$T' \to *FT'$
$\$E'T'F$	id$	
$\$E'T'$**id**	id$	$F \to$ **id**
$\$E'T'$	$	
$\$E'$	$	$T' \to \varepsilon$
$	$	$E' \to \varepsilon$

	输入符号					
	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \varepsilon$	$E' \to \varepsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \varepsilon$	$T' \to *FT'$		$T' \to \varepsilon$	$T' \to \varepsilon$
F	$F \to$ **id**			$F \to (E)$		

例: 预测分析器推导过程的顺序

$$
\begin{aligned}
E &\Rightarrow TE' \Rightarrow FT'E' \Rightarrow \mathbf{id}\ T'E' \Rightarrow \mathbf{id}\ E' \\
&\Rightarrow \mathbf{id} + TE' \Rightarrow \mathbf{id} + FT'E' \\
&\Rightarrow \mathbf{id} + \mathbf{id}\ T'E' \Rightarrow \mathbf{id} + \mathbf{id} * FT'E' \\
&\Rightarrow \mathbf{id} + \mathbf{id} * \mathbf{id}\ T'E' \\
&\Rightarrow \mathbf{id} + \mathbf{id} * \mathbf{id}\ E' \\
&\Rightarrow \mathbf{id} + \mathbf{id} * \mathbf{id}
\end{aligned}
$$

已经扫描过的输入符号加上栈中的文法符号(from top to bottom)构成该推导的**左句型**

龙书中FIRST, FOLLOW 的计算方法 (注意: 定义和虎书略有不同)

FIRST的计算方法

- **计算FIRST(X), $X \in T \cup N$**

  Base Case:

  X是终结符($X \in T$), FIRST(X) = $\{X\}$

  Inductive Case:

  1. X是非终结符且 $X \rightarrow \varepsilon$ 则将ε加入到FIRST(X)
  2. X是终结符 且 $X \rightarrow Y_1\ Y_2 \ldots Y_k$
     - 如果 $a \in$ FIRST(Y_i)且ε在FIRST(Y_1), …, FIRST(Y_{i-1})中，则将a加入到FIRST(X)
     - 如果ε 在FIRST(Y_1), …, FIRST(Y_k)中， 则将ε 加入到FIRST(X)

FIRST集只包括终结符和ε

FIRST的计算方法

- **计算FIRST(X), $X \in T \cup N$**
  - 如上页ppt..
- **计算FIRST($X_1X_2 \ldots X_n$)**
  - 加入FIRST(X_1)中所有非ε符号
  - 若ε在FIRST(X_1)中，加入FIRST(X2)中所有非ε符号 …
  - 若ε在所有FIRST(X_i)中，也加入ε

计算FOLLOW集

- **计算FOLLOW(A), $A \in N$**
  - \$加入到FOLLOW(A)，当A是开始符号，\$ 是输入串的结束符号
  - 按照下面两个规则不断迭代，直到所有的FOLLOW集合都不再增长为止
    - 如果$A \rightarrow \alpha B\beta$ ，则FIRST(β)-{ε}加入到FOLLOW(B)
    - 如果$A \rightarrow \alpha B$ 或$A \rightarrow \alpha B\beta$且$\varepsilon \in$ FIRST(β)，则FOLLOW(A)中所有符号加入到 FOLLOW(B)

例: FIRST集计算

表达式文法(无左递归的)

- **例** $E \to TE'$
  $E' \to +TE' \mid \varepsilon$
  $T \to FT'$
  $T' \to *FT' \mid \varepsilon$
  $F \to (E) \mid \mathbf{id}$

- $X \in T$, FIRST(X) = {X}
- $X \in N$且 $X \to \varepsilon$, $\varepsilon \in$ FIRST(X)
- $X \in N$ 且 $X \to Y_1\, Y_2 \ldots Y_k$
  - 如果 $a \in$ FIRST(Y_i)且ε在FIRST(Y_1), …, FIRST(Y_{i-1})中，则$a \in$ FIRST(X)
  - 如果 ε 在FIRST(Y_1), …, FIRST(Y_k)中，则$\varepsilon \in$ FIRST(X)

FIRST(F) = { (, id }

例: FIRST集计算

表达式文法(无左递归的)

- **例** $E \to TE'$
  $E' \to +TE' \mid \varepsilon$
  $T \to FT'$
  $T' \to *FT' \mid \varepsilon$
  $F \to (E) \mid \text{id}$

❑ $X \in \mathrm{T}$, FIRST(X) = $\{X\}$

❑ $X \in \mathrm{N}$ 且 $X \to \varepsilon$, $\varepsilon \in$ FIRST(X)

❑ $X \in \mathrm{N}$ 且 $X \to Y_1 Y_2 \ldots Y_k$

- ❖ 如果 $a \in$ FIRST(Y_i)且ε在FIRST(Y_1), …, FIRST(Y_{i-1})中，则$a \in$ FIRST(X)
- ❖ 如果 ε 在FIRST(Y_1), …, FIRST(Y_k)中，则$\varepsilon \in$ FIRST(X)

FIRST(F) = { (, id } = FIRST(T) = FIRST(E)

例: FIRST集计算

表达式文法(无左递归的)

- **例** $E \rightarrow TE'$

  $E' \rightarrow +TE' \mid \varepsilon$

  $T \rightarrow FT'$

  $T' \rightarrow *FT' \mid \varepsilon$

  $F \rightarrow (E) \mid \mathbf{id}$

- $X \in N$ 且 $X \rightarrow \varepsilon$, $\varepsilon \in$ FIRST(X)
- $X \in T$, FIRST(X) = {X}
- $X \in N$ 且 $X \rightarrow Y_1 Y_2 \ldots Y_k$
  - 如果 $a \in$ FIRST(Y_i)且ε在FIRST(Y_1), …, FIRST(Y_{i-1})中，则$a \in$ FIRST(X)
  - 如果 ε 在FIRST(Y_1), …, FIRST(Y_k)中，则$\varepsilon \in$ FIRST(X)

FIRST(F) = { (, id } = FIRST(T) = FIRST(E)

FIRST(E') = {+, ε}

例: FIRST集计算

表达式文法(无左递归的)

- **例** $E \rightarrow TE'$
  $E' \rightarrow + TE' \mid \varepsilon$
  $T \rightarrow FT'$
  $T' \rightarrow * FT' \mid \varepsilon$
  $F \rightarrow (E) \mid \mathbf{id}$

❑ $X \in T$, FIRST(X) = $\{X\}$

❑ $X \in N$且 $X \rightarrow \varepsilon$, $\varepsilon \in$ FIRST(X)

❑ $X \in N$ 且 $X \rightarrow Y_1\ Y_2 \ldots Y_k$

- ❖ 如果 $a \in$ FIRST(Y_i)且ε在FIRST(Y_1), …, FIRST(Y_{i-1})中，则$a \in$ FIRST(X)
- ❖ 如果 ε 在FIRST(Y_1), …, FIRST(Y_k)中，则$\varepsilon \in$ FIRST(X)

FIRST(F) = { (, id } = FIRST(T) = FIRST(E)

FIRST(E') = $\{+, \varepsilon\}$

FRIST(T') = $\{*, \varepsilon\}$

例: FIRST集计算

表达式文法(无左递归的)

- **例** $E \rightarrow TE'$
  $E' \rightarrow + TE' \mid \varepsilon$
  $T \rightarrow FT'$
  $T' \rightarrow * FT' \mid \varepsilon$
  $F \rightarrow (E) \mid \mathbf{id}$

❑ $X \in \mathrm{T}$, FIRST(X) = {X}

❑ $X \in \mathrm{N}$且 $X \rightarrow \varepsilon$, $\varepsilon \in$ FIRST(X)

❑ $X \in \mathrm{N}$ 且 $X \rightarrow Y_1\ Y_2 \ldots Y_k$

❖ 如果 $a \in$ FIRST(Y_i)且ε在FIRST(Y_1), …, FIRST(Y_{i-1})中，则$a \in$ FIRST(X)

❖ 如果 ε 在FIRST(Y_1), …, FIRST(Y_k)中，则$\varepsilon \in$ FIRST(X)

FIRST(E) = FIRST(T) = FIRST(F) = { (, id }
FIRST(E') = {+, ε}
FRIST(T') = {*, ε}

例: FOLLOW集计算

表达式文法(无左递归的)

- **例** $E \to TE'$
  $E' \to +TE' \mid \varepsilon$
  $T \to FT'$
  $T' \to *FT' \mid \varepsilon$
  $F \to (E) \mid \mathbf{id}$

- **当A是开始符号，$ ∈ FOLLOW(A)**
- $A \to \alpha B\beta$ ，$\mathrm{FIRST}(\beta)-\{\varepsilon\} \subseteq \mathrm{FOLLOW}(B)$
- $A \to \alpha B$ **或** $A \to \alpha B\beta$ **且** $\varepsilon \in \mathrm{FIRST}(\beta)$ ，
  $\mathrm{FOLLOW}(A) \subseteq \mathrm{FOLLOW}(B)$

$\mathrm{FIRST}(E) = \mathrm{FIRST}(T) = \mathrm{FIRST}(F) = \{\ (\ ,\ \mathbf{id}\ \}$
$\mathrm{FIRST}(E') = \{+, \varepsilon\}$
$\mathrm{FRIST}(T') = \{*, \varepsilon\}$
$\mathrm{FOLLOW}(E) = \{\ ),\ \$\}$

例: FOLLOW集计算

表达式文法(无左递归的)

- 例 $E \to TE'$
  $E' \to + TE' \mid \varepsilon$
  $T \to FT'$
  $T' \to * FT' \mid \varepsilon$
  $F \to (E) \mid \mathbf{id}$

❑ 当A是开始符号，$ \$ \in$ FOLLOW(A)

❑ $A \to \alpha B\beta$ ，FIRST(β)-{ε} $\subseteq$ FOLLOW(B)

❑ $A \to \alpha B$ 或$A \to \alpha B\beta$且$\varepsilon \in$ FIRST(β)，FOLLOW(A) $\subseteq$ FOLLOW(B)

FIRST(E) = FIRST(T) = FIRST(F) = { (, id }
FIRST(E') = {+, ε}
FRIST(T') = {*, ε}
FOLLOW(E) = {), \$} = FOLLOW(E')

例: FOLLOW集计算

表达式文法(无左递归的)

- **例** $E \to TE'$
  $E' \to +TE' \mid \varepsilon$
  $T \to FT'$
  $T' \to *FT' \mid \varepsilon$
  $F \to (E) \mid \mathbf{id}$

- 当A是开始符号，$\$ \in$ FOLLOW(A)
- $A \to \alpha B\beta$，FIRST(β)-{ε} $\subseteq$ FOLLOW(B)
- $A \to \alpha B$ 或 $A \to \alpha B\beta$ 且 $\varepsilon \in$ FIRST(β)，FOLLOW(A) $\subseteq$ FOLLOW(B)

FIRST(E) = FIRST(T) = FIRST(F) = { (, id }
FIRST(E') = {+, ε}
FRIST(T') = {*, ε}
FOLLOW(E) = {), \$} = FOLLOW(E')
FOLLOW(T) = {+,), \$}

例: FOLLOW集计算

表达式文法(无左递归的)

- **例** $E \to TE'$
  $E' \to +TE' \mid \varepsilon$
  $T \to FT'$
  $T' \to *FT' \mid \varepsilon$
  $F \to (E) \mid \mathbf{id}$

- 当A是开始符号, $\$ \in$ FOLLOW(A)
- $A \to \alpha B\beta$ ，FIRST(β)-{ε} $\subseteq$ FOLLOW(B)
- $A \to \alpha B$ 或 $A \to \alpha B\beta$ 且 $\varepsilon \in$ FIRST(β)，FOLLOW(A) $\subseteq$ FOLLOW(B)

FIRST(E) = FIRST(T) = FIRST(F) = { (, id }
FIRST(E') = {+, ε}
FRIST(T') = {*, ε}
FOLLOW(E) = {), \$} = FOLLOW(E')
FOLLOW(T) = {+,), \$} = FOLLOW (T')

例: FOLLOW集计算

表达式文法(无左递归的)

- **例** $E \to TE'$
  $E' \to +TE' \mid \varepsilon$
  $T \to FT'$
  $T' \to *FT' \mid \varepsilon$
  $F \to (E) \mid \mathbf{id}$

❑ 当A是开始符号, $\$ \in$ FOLLOW(A)

❑ $A \to \alpha B\beta$ ，FIRST(β)-{ε} $\subseteq$ FOLLOW(B)

❑ $A \to \alpha B$ 或 $A \to \alpha B\beta$ 且 $\varepsilon \in$ FIRST(β) ，FOLLOW(A) $\subseteq$ FOLLOW(B)

FIRST(E) = FIRST(T) = FIRST(F) = { (, id }
FIRST(E') = {+, ε}
FRIST(T') = {*, ε}
FOLLOW(E) = {), \$} = FOLLOW(E')
FOLLOW(T) = {+,), \$} = FOLLOW (T')
FOLLOW(F) = {*,+,), \$}

例: FOLLOW集计算

- 例 $E \rightarrow TE'$

  $E' \rightarrow +TE' \mid \varepsilon$

  $T \rightarrow FT'$

  $T' \rightarrow *FT' \mid \varepsilon$

  $F \rightarrow (E) \mid \mathbf{id}$

FIRST(E) = FIRST(T) = FIRST(F) = { (, id } FIRST(E') = {+, ε}

FRIST(T') = {*, ε}

FOLLOW(E) = FOLLOW(E') = {), $}

FOLLOW(T) = FOLLOW (T') = {+,), $}

FOLLOW(F) = {+, *,), $}