第三章：语法分析

自顶向下语法分析概述
三个重要的集合

1. 自顶向下语法分析概述

- 自顶向下语法分析方法，亦称面向目标的分析方法。
  - 思想：从文法的开始符出发企图用最左推导推导出与输入的单词串完全相匹配的句子。若输入的单词串是给定文法的句子，则必能推出，反之必然以推导失败而终止。
  - 自顶向下语法分析方法分为：

    1、非确定的自顶向下语法分析方法；

    2、确定的自顶向下语法分析方法：实现方法简单、直观，便于手工构造和自动生成，是目前常用的语法分析方法。
    - 递归下降方法；
    - LL（1）方法。

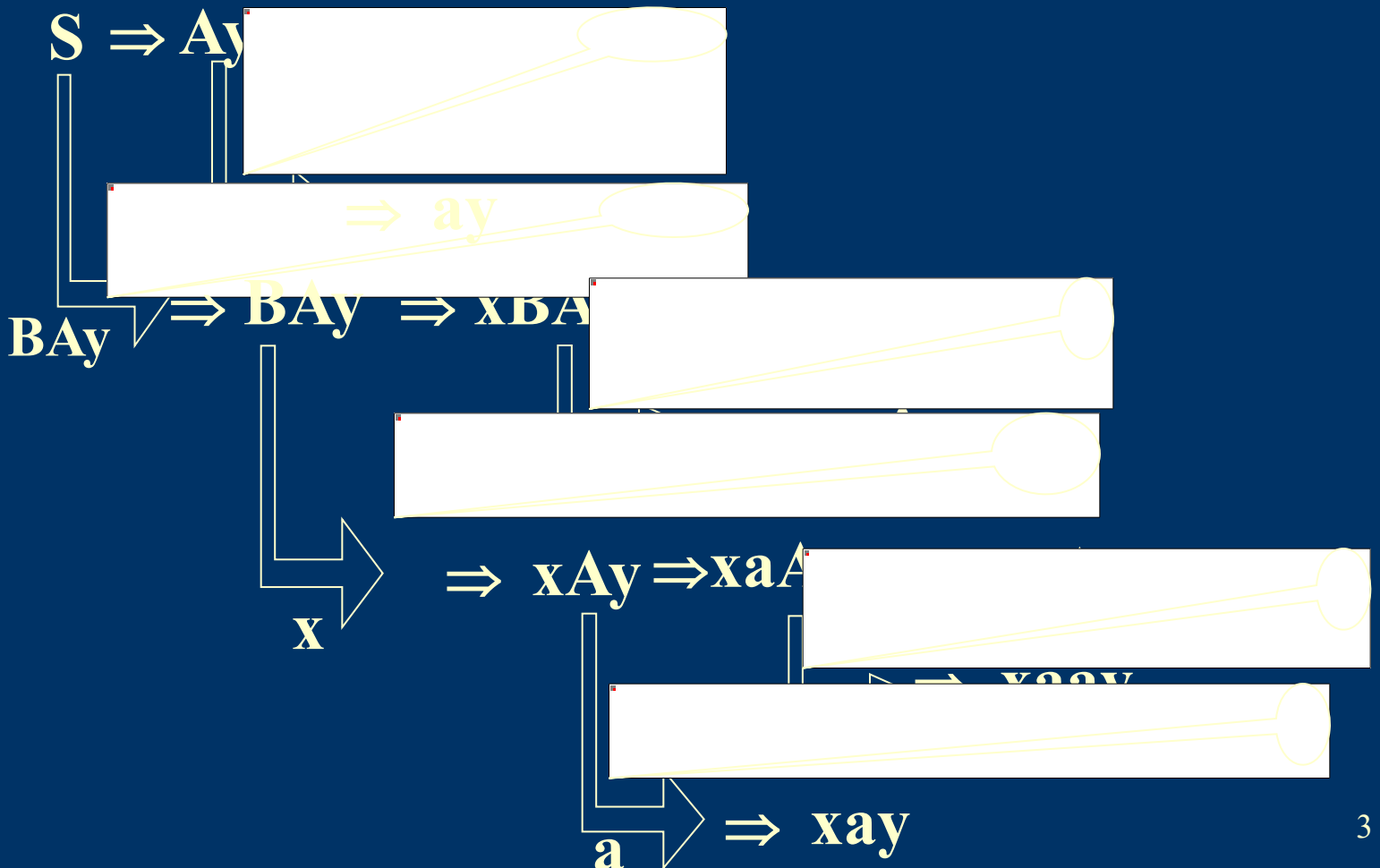
非确定的自顶向下语法分析方法（带回溯过程的自顶向下语法分析方法）

已知G[S]:

S→Ay | BAy

A→aA | a

B→xB | x 输入串为xay是否正确。



1. 自顶向下语法分析概述

- 选择规则的策略
  - 穷举的方法效率非常的差
  - 考虑更多的信息，如输入流
  - 根据输入流选取规则
  - 考察输入流中的几个符号

2.1 First集的定义

- 设$G=(V_T, V_N, S, P)$是上下文无关文法，

  $\beta \in (V_T \cup V_N)^*$

  $First(\beta)=\{ a \in V_T \mid \beta \Rightarrow^* a... \}$

  $\cup (if\ \beta \Rightarrow^* \varepsilon\ then\ \{\varepsilon\}\ else\ \varnothing)$

2.2 Follow集的定义

设$G=(V_T, V_N, S, P)$是上下文无关文法，
$A \in V_N$，S是开始符号

$$Follow(A)=\{\ a\ \in V_T \mid S \Rightarrow^{+} \ldots Aa \ldots\ \}$$
$$\cup(\text{if } S \Rightarrow^{*} \ldots A \text{ then } \{\#\} \text{ else } \varnothing)$$

2.3 Predict集的定义

Predict(A→β)

= First(β) , 当First(β)不含ε

= First(β)-{ε} ∪ Follow(A) ,

当First(β)含ε

3.1 计算First(X)集

- 若$X \in V_T$, First(X)={X}
- 若$X \in V_N$则

  First(X)={a| $X \to a\cdots \in P, a \in V_T$}
- 若$X \in V_N$, 且有产生式$X \to \varepsilon$, 则 $\varepsilon \in$First(X)
- 若$X \in V_N$, 有产生式$X \to Y_1Y_2\cdots Y_n$， 且$Y_1, Y_2, \cdots, Y_i \in V_N$

  当$Y_1, Y_2, \cdots, Y_{i-1} \Rightarrow^* \varepsilon$， Y_i 不能 $\Rightarrow^* \varepsilon$

  则First(Y_1)-{ε}, First(Y_2)-{ε}, …

  First(Y_{i-1})-{ε}， First(Y_i)都包含在First(X)中。

  当所有$Y_i \Rightarrow^* \varepsilon$(i=1, 2, …n)， 则将所有First($Y_i$)并入First(X) 中。

3.2 计算First(α)集

设符号串$\alpha=X_1X_2\cdots X_n$，
当$X_1,X_2,\ldots,X_{i-1}\Rightarrow^*\varepsilon$，$X_i$不能$\Rightarrow^*\varepsilon$，
则：

$$First(\alpha)=\cup_1^{i-1}(First(X_j)-\{\varepsilon\})\cup First(X_i)$$

- 若所有X_i都能$\Rightarrow^*\varepsilon$，

则：

$$First(\alpha)=\cup_1^{n} First(X_j)$$

例：

- E → T E'
- E' → + T E' | ε
- T → F T'
- T' → * F T' | ε
- F → id | (E)

文法符号	First集
E	{ id, (}
E'	{ +, ε}
T	{ id, (}
T'	{ *, ε }
F	{ id, (}

符号串	First集
T E'	{ id, (}
F T'	{ id, (}
E' T' T	{ +, *, id, (}
E' T'	{ +, *, ε }

4. 计算Follow集

1: 对所有$A \in V_N$ 且 A非开始符：

令Follow(A):={ };

对开始符 S：令Follow(S)={ # };

2: 若有产生式A→xBy：

如果ε ∈First(y) 则：

Follow(B)

= Follow(B) ∪ (First(y)－{ε}) ∪ Follow(A)

否则：

Follow(B)

= Follow(B) ∪ First(y)

3: 重复2，直至对所有$A \in V_N$，Follow(A)收敛为止。

文法符号	First集	Follow集
E	{ id, (}	{# ,) }
E’	{ +, ε}	{ # ,) }
T	{ id, (}	{ + , # ,) }
T’	{ *, ε }	{+ , # ,) }
F	{ id, (}	{* , + , # ,)}

例：

① E → T E’

② E’ → + T E’

③ | ε

④ T → F T’

⑤ T’ → * F T’

⑥ | ε

⑦ F → id

⑧ | (E)

5. 计算Predict集

Predict(A→β)

$$= \begin{cases} \text{First}(\beta), & \text{当First}(\beta)\text{不含}\varepsilon \\ (\text{First}(\beta)-\{\varepsilon\}) \cup \text{Follow}(A), & \text{当First}(\beta)\text{含}\varepsilon \end{cases}$$

例：

- E → T E'
- E' → + T E' | ε
- T → F T'
- T' → * F T' | ε
- F → id | (E)

文法符号	First集	Follow集
E	{ id, (}	{), # }
E'	{ +, ε}	{), # }
T	{ id, (}	{ +,), # }
T'	{ *, ε }	{ +,), # }
F	{ id, (}	{ *, +,), # }

Predict(E→TE') = first(TE') = { id , (}

Predict(E' →+TE') = first(+TE') = { + }

Predict(E' → ε) = follow(E') = {) , # }

Predict(T →FT') = first(FT') = { id , (}

例：

- E → T E'
- E' → + T E' | ε
- T → F T'
- T' → * F T' | ε
- F → id | (E)

文法符号	First集	Follow集
E	{ id, (}	{), # }
E'	{ +, ε}	{), # }
T	{ id, (}	{ +,), # }
T'	{ *, ε }	{ +,), # }
F	{ id, (}	{ *, +,), # }

Predict(T' →*FT') = first(*FT') = { * }

Predict(T' → ε) = follow(T') = { + ,) , # }

Predict(F →id) = first(id) = { id }

Predict(F →(E)) = first((E)) = { (}

- 非确定的自顶向下语法分析方法主要问题：虚假匹配而引起回溯，原因是：

1.由于相同左部的产生式的右部的FIRST集的交集不为空而引起回溯。

$First(cAe) \cap First(cAd) \neq \varnothing$

例：有如下文法G[Z]：

Z→ cAe | cAd

A → eb | a

$Predict(Z \rightarrow cAe) \cap Predict(Z \rightarrow cAd) \neq \varnothing$

分析字符串cad 。

■至多有一个产生式被选择的条件是：

$predict(A \rightarrow \beta_k) \cap predict(A \rightarrow \beta_j) = \varnothing$，当$k \neq j$

2.由于某非终极符的产生式的右部能推导出ε，且该非终极符的FOLLOW集中含有其某个产生式右部的FIRST集中的元素。

例：有如下文法G[S]：

$S \rightarrow aAS \mid b$

$A \rightarrow bAS \mid \varepsilon \mid d$

分析字符串ab 。

$First(bAS) \cap First(\varepsilon) = \varnothing$

$First(bAS) \cap First(d) = \varnothing$

$First(d) \cap First(\varepsilon) = \varnothing$

$Follow(A) = \{b,a\}$

$Predict(A \rightarrow bAS) \cap Predict(A \rightarrow \varepsilon) \neq \varnothing$

■至多有一个产生式被选择的条件是：

$predict(A \rightarrow \beta_k) \cap predict(A \rightarrow \beta_j) = \varnothing$，当$k \neq j$

满足该条件的文法称为LL(1)文法，递归下降子程序文法。

- 非LL(1)文法到LL(1)文法的等价转换
  - 消除左公共前缀
  - 消除直接左递归
  - 消除左递归

注意1： 即使文法没有公共前缀和左递归也并不意味着文法一定是LL(1)文法，有时还需要其它的转换方法。

注意2： 通常程序设计语言的文法大都可以转换成LL(1)文法，但已经证明，非LL(1)文法是存在的，即有些文法是无法变换成LL(1)文法的。

类型题

	First集	Follow集	Predict集
S → ABc	{a,b,c}	S={#}	{a,b,c}
A → a	{a}	A={b,c}	{a}
A → ε	{ε}		{b,c}
B→ b	{b}	B={c}	{b}
B → ε	{ε}		{c}