求 FIRST集，FOLLOW集，FIRSTVT 集和 LASTVT 集

技术标签： 编译原理

FIRST 集的定义

简单的说 FIRST 集就是一个文法符号串的开始符号集合

设 G=（VT，VN，S，P）是上下文无关文法，

FIRST(α)={a|α=>aβ,a∈VT,α,β∈v\*}

若 α=> ε（经过0或多步推导可以推出为空串），则规定 ε ∈ FIRST(α)

FIRST(α) 是 α 的所有可能推导的开头终结符或可能的 ε。

例题 4.3：求 FIRST 集

题目：

给定文法 G[S]：

（1）S -> Af

（2）S -> Be

（3）A -> a

（4）A -> cA

（5）B -> b

（6）B -> dB

详解：

求（1）中的 Af 的 FIRST 集，注意，因为如果推出为空时用 ε，所以 A 后面的 f 是没用的，我们只分析 A 的第一个终结符的集。

因为（3）和（4）都是由 A 推导，所以两个都考虑

FIRST(Af) = FIRST(a) ∪ FIRST(cA) = {a,c}

同理可求出：

FIRST(Be)

FIRST(a)

FIRST(cA)

FIRST(dB)

（三）关于 FOLLOW 集 - 后随集

如果仅适用 FIRST 只能根据首字符不同选择产生式，如果首字符不同…

FOLLOW 集的定义

简单的说 FOLLOW 集就是一个文法符号的后跟终结符号的集合。

设 G =（VT，VN，S，P）是上下文无关文法，A∈VN，S是开始符号。

FOLLOW(A)={a|S=>\*…Aa…，a ∈VT}

若有 S=>\*…A（就是说 A 已经是最后一个时，没有后面的），则规定 # ∈ FOLLOW(A)

FOLLOW(A) 是所有出现在紧接 A 之后的终结符或 “#”；

FOLLOW 计算规则

（1） 对于文法的开始符号 S，置 # 到 FOLLOW(S) 中；

（2）若 A -> αBaβ 是一个产生式，a 为终结符，则把 a 加至 FOLLOW(B) 中；

（3）若 A -> αBβ 是一个产生式，则把 FIRST(β) - {ε} 加至 FOLLOW(B) 中；

（4）若 A -> αB 是一个产生式，或 A -> αBβ 是一个产生式，而 β =\*> ε，

则把 FOLLOW(A) 加至 FOLLOW(B) 中

提示：

（1）就是说如果对开始符号求 FOLLOW(S) ，直接来个 # ∈FOLLOW(S) ，不过要表示成 {#}

（2）就是把后面的紧跟的终结符，就直接加到 FOLLOW 集

（3）正经的求 B 的 FOLLOW 集，就是 B 后面 β 的 FIRST(β) - {ε}

（4）分情况：

如果 A -> αB，就把 FOLLOW(A) 加至 FOLLOW(B) 中

A -> αBβ 是一个产生式，此时 β 可以推成 ε，就是相当于也能推出 A -> αB，也把 FOLLOW(A) 加至 FOLLOW(B) 中

注意： （4）中这里是 FOLLOW(A) 加至 FOLLOW(B) ，就是左部的 FOLLOW 集，加到其推导出的右部的最后一个非终结符的 FOLLOW 集，

例如：需要识别输入符号串 ：bsAcD，求 FOLLOW(B) 的时候

此时 FOLLOW(A) 就含有 c，如果 A -> dB，即此时 FOLLOW(B) 也应该有 c

记忆方式：

提示：这是规则，不是求某个固定谁的 FOLLOW 集，而涉及多个非终结符的 FOLLOW 集，所以建议对每个产生式对这 4 个规则都要考虑，不然很容易漏。

规则（1）看左侧为开始符；

规则（2）右侧看 B 后是否紧跟终结符；

规则（3）右侧看 B 后紧跟的是否有非终结符

规则（4）右侧看 B 是不是最后一个，或 B 后面的可以推出空串，间接最后一个

例题 4.4：求 FOLLOW 集

题目：

给定文法 G[S]：

（1）S -> eT|RT

（2）T -> DR|ε

（3）R -> dR|ε

（4）D -> a|bd

详解：

计算时，要同时考虑四个规则是否满足，就是都要考虑

对产生式（1）：

1.满足规则（1），因为 S 是开始符号，可以得到 FOLLOW(S) = {#}

2.不满足规则（2）

3.满足规则（3），对 S -> RT，应把 FIRST(T) - {ε} = {a,b} 加到 FOLLOW(R)；

4.满足规则（4），将 FOLLOW(S)={#} 加到 FOLLOW(T)

对产生式（2）：

1.不满足规则（1）

2.不满足规则（2）

3.满足规则（3），对 T -> DR，应把 FIRST® - {ε} = {d} 加到 FOLLOW(D)；

4.满足规则（4），将 FOLLOW(T)={#} 加到 FOLLOW(R)

对产生式（3）：

1.不满足规则（1）

2.不满足规则（2）

3.不满足规则（3）

4.不满足规则（4），前后 R 和 R 一样不用加

对产生式（4）：

1.不满足规则（1）

2.不满足规则（2）

3.不满足规则（3）

4.不满足规则（4）

最终结果：

FOLLOW(S) = {#}

FOLLOW(T) = {#}

FOLLOW(R) = {a, b, #}

FOLLOW(D) = {d, #}

FIRSTVT(T)

LASTVT(T)

表示及含义：

FIRSTVT(T) 非终结符T的最左终结符集合

LASTVT(T) 非终结符T的最右终结符集合

定义：

定义解释：

FIRSTVT(T) 非终结符T经过1步或多步推导，得到的最左端终结符，以及左端第二个终结符的集合

LASTVT(T) 非终结符T经过1步或多步推导，得到的最右端终结符，以及倒数第二个终结符的集合

求 FIRSTVT 集的步骤：

（1）若有产生式 T→a 或者 T→Ra…，则 a ∈ FIRSTVT(T)

（2）若 a ∈ FIRSTVT®，且有产生式 T→R…，则 a ∈ FIRSTVT(T)

​ 就是说如果 a 是非终结符 R 的 FIRSTVT 集，且 T 可以推出以非终结 R 带头的右部，则 a 也是非终结符 T 的 FIRSTVT 集。

​ 注： 省略号 … 可以为空，就是没有

求 LASTVT 集的步骤：

（1）若有产生式 T→…a 或者 T→…aR，则 a ∈ LASTVT(T)

（2）若 a ∈ LASTVT®，且有产生式 T→…R，则 a ∈ LASTVT(T)

例题：

已给文法：

G[S]:

S→a|b|(B)

A→S, A|S

B→A

求所有非终结符的 FIRSTVT，LASTVT 集

解析：

（1）只要是让求 FIRSTVT，LASTVT 集，则该文法就隐含条件为算符优先文法。

（2）算符优先文法的特点是：不会出现两个相邻的非终结符，即两个非终结符中间夹着一个终结符。如果第一个是终结符则第二个是非终结符。

结果：

FIRSTVT 集 LASTVT 集

S {a, b, ( } {a, b, ) }

A {a, b, (, 逗号 } {a, b, ), 逗号}

B {a, b, (, 逗号 } {a, b, ), 逗号}

