4.5 自底向上的语法分析→∪Σεαβ∈∉

LL(1)文法:

对递归分析法: A() {}

对非递归分析法: Move(A, a)

LL(1)文法, 判断条件:

任何两个产生式A → α | β 都满足下列条件：

1. FIRST(α) ∩ FIRST(β ) = Φ
2. 若β ==>\* ε，那么FIRST(α) ∩ FOLLOW(β) = Φ

构造预测分析表:

对于一个文法*G*的每个产生式*A*→α，进行如下处理：

* + 对于FIRST(α)中的每个终结符号a，将*A*→α加入到*M*[*A*,a]。
  + 如果ε在FIRST(α)中，那么对于FOLLOW(*A*)中的每个终结符号b，将*A*→α加入到*M*[*A*,b]中。如果ε在FIRST(α)中，且$在FOLLOW(*A*)中，也将*A*→α加入到*M*[*A*,$]。

自顶向下 E => E + E 🡺 id \* id + id

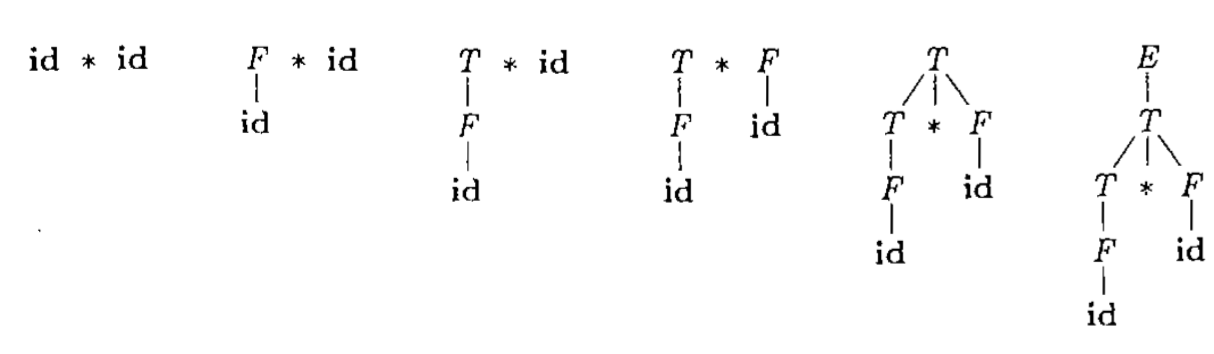
*E → E + T | T*

*T → T \* F | F*

*F* → (*E* ) | **id**

自底向上语法分析

id \* id



自底向上语法分析:

一个自底向上语法分析过程对应于为

一个输入串构造语法分析树的过程，

它从叶子节点开始，逐步向上到达根节点。

即把输入串**归约**成文法的开始符号。

归约:

若一个子串和某个产生式的右部匹配，

则用该产生式的左部符号代替这个子串。

例(1) 写出如下文法及分析串 abbcde的自底向上分析过程

*S* → *aABe*

*A → Abc | b*

*B → d*

待分析串: abbcde

abbcde 🡺 aAbcde 🡺 aAde 🡺 aABe 🡺 S

S ⇒rm aABe ⇒rm aAde ⇒rm aAbcde ⇒rm abbcde

最右推导的逆过程

对输入进行从左到右的扫描，

并在扫描过程中进行自底向上语法分析，

就可以反向构造出一个最右推导。

句柄(handle)

句型: 由文法开始符号S,可以推导出来的终极符号或者非终极符号构成的串

句型的句柄是和某产生式右部匹配的子串，

并且，把它归约成该产生式左部的非终结符

代表了最右推导过程的逆过程的一步。

S → aABe

A → Abc | b

B → d

S ⇒rm aABe ⇒rm aAde ⇒rm aAbcde ⇒rm abbcde

S ⇒rm**aABe** ⇒rm aA**d**e ⇒rm a**Abc**de ⇒rm a**b**bcde

句柄的**右边仅含终结符;**

如果文法二义, 那么句柄可能不唯一.

句柄不唯一

E → E \* E | E + E | id id1 \* id2 + id3

E ⇒rm E \* E

⇒rm E \* E + E

⇒rm E \* E + id3

⇒rm E \* id2 + id3

⇒rm id1 \* id2 + id3

E ⇒rm E + E

⇒rm E + id3

⇒rm E \* E + id3

⇒rm E \* id2 + id3

⇒rm id1 \* id2 + id3

句型 E \* E + id 的句柄

移进-归约语法分析是自底向上语法分析的一种形式。

使用栈来保存文法符号，并用一个输入缓冲区来存放将要进行语法分析的其余符号。

移进-归约语法分析的四种动作：

* 移进 (shift)：把下一个输入符号移进栈 (栈顶)
* 归约 (reduce)：分析器知道句柄的右端已在栈顶，然后确定句柄的左端在栈中的位置，再决定用什么样的非终结符代替句柄
* 接受 (accept)：分析器宣告分析成功
* 报错 (error)：分析器发现语法错误，调用错误恢复例程

*E → E + T | T*

*T → T \* F | F*

*F* → (*E* ) | **id**

|  |  |  |
| --- | --- | --- |
| 栈 | 输入 | 动作 |
| $ | id1 \* id2 $ | 移进 |
| $ id1 | \* id2 $ | 归约 F 🡪 id |
| $ F | \* id2 $ | 归约 T 🡪 F |
| $ T | \* id2$ | 移进 |
| $ T \* | id2 $ | 移进 |
| $ T \* id2 | $ | 归约 F 🡪 id |
| $ T \* F | $ | 归约 T 🡪 T \* F |
| $T | $ | 归约 E 🡪 T |
| $ E | $ | 接受 |

之所以能够在移入-归约语法分析中使用栈，

是因为这个分析过程具有如下重要性质：

句柄总是出现在栈的顶端，绝不会出现在栈的中间。

要想很好地使用移进−归约方式，尚需解决一些问题:

* + 如何决策选择移进还是归约
  + 进行归约时，确定句型中将要归约的子串
  + 进行归约时，如何确定选择哪一个产生式

例(2) 移进-归约冲突

stmt → **if** expr **then** stmt

| **if** expr **then** stmt **else** stmt

| other

|  |  |  |
| --- | --- | --- |
| 栈 | 输入 | 动作 |
| **if** expr **then** stmt | **else** …. $ | ?? |

例(3) 归约-归约冲突

stmt → id (parameter\_list) | expr = expr

parameter\_list → parameter\_list, parameter | parameter

parameter → id

expr → id (expr\_list) | id

expr\_list → expr\_list, expr | expr

|  |  |  |
| --- | --- | --- |
| 栈 | 输入 | 动作 |
| … id ( id | , id ) … | ?? |

练习 1: S → S S + | S S \* | a

求 SSS + a \* +

SS + a \* a +

aaa \* a ++

以上句型的句柄.

练习2: 对于下面的输入符号串和文法，

说明相应的自底向上语法分析过程:

1. 文法S → 0 S 1 | 0 1 的串 000111
2. 文法 S → S S + | S S \* | a的串aaa \* a ++

|  |  |  |
| --- | --- | --- |
| 栈 | 输入 | 动作 |
| $ | 000111$ | 移进 |
| $0 | 00111$ | 移进 |
| $00 | 0111$ | 移进 |
| $000 | 111$ | 移进 |
| $0001 | 11$ | 归约 S 🡪 01 |
| $00S | 11$ | 移进 |
| $00S1 | 1$ | 归约 S 🡪 0S1 |
| $0S | 1$ | 移进 |
| $0S1 | $ | 归约 S 🡪 0S1 |
| $S | $ | 接受 |

|  |  |  |
| --- | --- | --- |
| 栈 | 输入 | 动作 |
| $ | aaa \* a ++$ | ?? |
| … | … | … |
| $ S | $ | accept |