

语法分析

上下文无关文法

自顶向下

FIRST集和FOLLOW集

FIRST集

FOLLOW集

LL(1)文法

递归分析

非递归分析

自底向上

LR(k)分析

有穷状态自动机

LR(0)

SLR(1)

LR(1)

LALR(1)

语法分析

上下文无关文法

使用上下文无关文法进行语法分析(终结符和非终结符的任意组合)

概念: 句柄、直接短语、短语、分析树(p23、p25)

文法变换:

- 消除二义性: 使用优先级
- 消除左递归: p29
- 提取左公因子: p31

自顶向下

FIRST集和FOLLOW集

FIRST集

- 若 $X \in$ 终结符

$$FIRST(X) = X$$

- 若 $X \in$ 非终结符

若 $X \rightarrow Y_1 Y_2$ 且 $Y_1 \rightarrow \epsilon$
则将 $FIRST(Y_2)$ 加入到 $FIRST(X)$ 中

若 $X \rightarrow \epsilon$
则将 ϵ 加入到 $FIRST(X)$ 中

FOLLOW集

- 若 S 是开始符号($\$$ 是右端结束标记)

将 $\$$ 放入 $FOLLOW(S)$ 中

- 若存在产生式 $A \rightarrow \alpha B \beta$

将 $FIRST(\beta)$ 中除 ϵ 之外的所有符号放入 $FOLLOW(B)$ 中

- 若存在产生式 $A \rightarrow \alpha B$ 或存在产生式 $A \rightarrow \alpha B \beta$ 且 $\epsilon \in FIRST(\beta)$

将 $FOLLOW(A)$ 中所有符号放入 $FOLLOW(B)$ 中

LL(1)文法

- 第一个L表示从左向右扫描输入
- 第二个L表示最左推导,
- 1表示每一步中只需要向前看一个输入符号来决定语法分析动作

对于 $A \rightarrow \alpha | \beta$ 任意两个产生式

- 不存在终结符号 a 使得 α 和 β 都可以推导出以 a 开头的串

$$FIRST(\alpha) \cap FIRST(\beta) = \epsilon$$

- α 和 β 最多只有一个可以推导出空串
- 如果 $\beta \Rightarrow \epsilon$, 则 α 不能推导出任何以 $FOLLOW(A)$ 中某个终结符号开头的串

$$FOLLOW(A) \cap FOLLOW(\alpha) = \epsilon$$

递归分析

由一组过程组成, 每个非终结符号有一个对应的过程

- 可能需要回溯
- 无法处理左递归文法(进入无限循环)

具体实现: 函数递归调用(p87)

非递归分析

非递归的预测分析器, 由五部分组成:

- 输入缓冲区: 存放被分析的输入符号串, 串尾的符号 $\$$ 是符号串结束标志
- 输出流: 分析过程中所采用的产生式序列
- 分析栈: 存放一系列文法符号, 符号 $\$$ 标识栈底。开始分析时, 先将 $\$$ 压入栈, 再将开始符号压入
- 预测分析程序: 根据栈顶符号 X 和当前输入符号 a 来决定分析程序应采取的动作
 - 若 $X = a = \$$, 分析成功, 停止分析
 - 若 $X = a \neq \$$, 从栈顶弹出 X , 向前指针前移一个位置

- 若 X 是终结符号, 且 $X \neq a$, 发生错误
- 若 X 是终结符号, 则访问预测分析表 $M[X, a]$
 - 若 $M[X, a]$ 是产生式 $X \rightarrow Y_1 Y_2 \dots Y_k$, 则先将 X 弹出栈顶, 然后将产生右部反序($Y_k \dots Y_2 Y_1$)压入栈中
 - 若 $M[X, a]$ 是产生式 $X \rightarrow \epsilon$, 则预测分析程序从栈顶弹出 X
- 预测分析表

如有当前输入符号 a , 产生式 $A \rightarrow \beta$, 当 A 位于分析栈栈顶时:

- 如果当前输入符号 $a \in FIRST(\beta)$, 表项 $M[A, a]$ 应放入产生式 $A \rightarrow \beta$
- 对应LL(1)文法规则1**
- 如果 $\epsilon \in FIRST(\beta)$, 且当前输入符号 $a \in FOLLOW(A)$, 表项 $M[A, a]$ 应放入产生式 $A \rightarrow \beta$

对应LL(1)文法规则2

自底向上

LR(k)分析

LR(k)分析方法:

- L表示自左向右扫描字符串
- R表示为输入字符串构造一个最右推导的逆过程(关键在于找到当前句型的句柄)
- k表示需要向前看的输入符号的个数

基本思想是:

- 一方面要记住历史信息, 即已经移进和归约出的整个字符串
- 另一方面要预测未来, 即根据所用的推测未来可能遇到的输入符号

一个LR分析程序包括五部分:

- 输入缓冲区: 存放待分析的输入字符串, 以 $\$$ 作为字符串的结束标志
 - 输出: 由LR分析控制程序分析输入字符串的过程中所采用的动作序列
 - 栈: 由状态栈和符号栈构成, 两个栈同步增减
 - 状态栈存放形如 $S_0 S_1 \dots S_m$ 的状态符号串, 栈底的 S_0 是初始状态, 栈顶的 S_m 是当前状态
每个状态符号概括了在栈中位于它下面的部分所包含的全部信息, 即从分析开始到某一归约阶段的全部历史信息 and 预测信息
 - 符号栈中存放形如 $X_0 X_1 \dots X_m$ 的文法符号串, 是从初态 S_0 到当前状态 S_m 的路径上各边的标记
- 有状态栈足够, 一般忽略符号栈**
- 分析表: 实际上是一个确定有限自动机的状态转移表。每一行对应一个状态, 每一列对应一个文法符号或者 $\$$ 。

其中终结符号及 $\$$ 对应的列构成**动作(action)表**, 非终结符号对应的列构成**状态转移(goto)表**, 即:

- $goto[S_m, A]$ 保存了当前状态 S_m 相对于非终结符 A 的后继状态
- $action[S_m, a_i]$ 规定了当前状态 S_m 面临输入符号 a_i 时应采取的分析动作, 包括:

- 移进：把当前输入符号 a_i 及 $S = goto[S_m, a_i]$ 分别压入符号栈和状态栈的栈顶，向前指针前移，指向下一个输入符号
- 归约：用产生式 $A \rightarrow \beta$ 进行归约,若 $|\beta| = r$ ，则分别从状态栈和符号栈的栈顶弹出 r 项，是 S_{m-r} 成为栈顶状态
然后把文法符号 A 及状态 $S = goto[S_{m-r}, A]$ 分别压入符号栈和状态栈的栈顶
- 接收：分析成功，停止分析
- 出错：调用出错处理程序，进行错误处理和恢复

- 分析控制程序：根据栈顶状态符号和向前指针所指向的符号(即当前输入符号)查找分析表，从中取得动作指示信息并采取相应的分析动作

活前缀：对于规范句型的一个前缀，如果它不包含句柄之后的任何符号，则称该前缀为该句型的一个活前缀。

它们是一个或若干个规范句型的前缀，即在其右边增加某些不同的终结符号串之后，就可以构成不同的规范句型。

有穷状态自动机

增广文法：添加 $S' \rightarrow S$ ，用于识别终止状态

项、项集(内核项、非内核项)、项集的闭包(龙书p154)

通过项集可构造一个有穷状态自动机，每个状态代表一个项集

GOTO函数：即有穷状态自动机的状态转移函数

项集 I 中 $[A \rightarrow \alpha \cdot X\beta]$ 到项集 J 中 $[A \rightarrow \alpha X \cdot \beta]$ 的转移：

$$GOTO(I, X) = J$$

语法分析表**ACTION函数**：

- 如果 $[A \rightarrow \alpha \cdot b\beta]$ 在 I_i 中(b 为终结符号)，并且 $GOTO(I_i, b) = I_j$ ，那么将 $ACTION[i, b]$ 设置为“移入 j ”(s j)
- 如果 $[A \rightarrow \alpha \cdot]$ 在 I_i 中，那么对于 $FOLLOW(A)$ 中的所有 b ，将 $ACTION[i, b]$ 设置为“归约 $A \rightarrow \alpha$ ”(r j)
- 如果 $[S' \rightarrow S \cdot]$ 在 I_i 中，那么将 $ACTION[i, \$]$ 设置为“接受”(acc)

语法分析表**GOTO函数**：

如果 $GOTO(I_i, A) = I_j$ (A 为非终结符)，那么 $GOTO[i, A] = j$

上述构造算法为SLR(1)分析表的构造。

对于LR(k)分析方法，区别主要在于 k 的值，即向前看的符号数，因而决定采取什么动作(移入还是归约)。

分析表的差异主要在于构造ACTION函数的归约动作

LR(0)

0表示向前看0个符号，即遇到终结符归约

ACTION函数：如果 $[A \rightarrow \alpha \cdot]$ 在 I_i 中，那么对于任意终结符 b ，将 $ACTION[i, b]$ 设置为“归约 $A \rightarrow \alpha$ ”(r j)

SLR(1)

1表示向前看1个符号，即遇到 $FOLLOW(A)$ 中的符号 b ，就归约

ACTION函数：如果 $[A \rightarrow \alpha \cdot]$ 在 I_i 中，那么对于 $FOLLOW(A)$ 中的所有 b ，将 $ACTION[i, b]$ 设置为“归约 $A \rightarrow \alpha$ ”(r j)

LR(1)

1表示向前看1个符号，只有遇到特定符号 b 才归约(看龙书p167)

- 在形如 $[A \rightarrow \alpha \cdot \beta, b]$ 且 $\beta \neq \epsilon$ 的项中，向前看符号没有任何作用
- 在形如 $[A \rightarrow \alpha \cdot, b]$ 的项中，只有下一个符号等于 b 时才要求按照 $A \rightarrow \alpha$ 进行归约

因此，区别在于ACTION函数归约动作的条件

ACTION函数：如果 $[A \rightarrow \alpha \cdot, b]$ 在 I_i 中且 $A \neq S'$ ，那么将 $ACTION[i, b]$ 设置为“归约 $A \rightarrow \alpha$ ”(r j)

该条件在于求项集的闭包时，特定符号的构造：

$for(I \text{ 中的每个项 } [A \rightarrow \alpha \cdot B\beta, b])$
 $for(\text{文法 } G' \text{ 中的每个产生式 } B \rightarrow \gamma)$
 $for(FIRST(\beta b) \text{ 中的每个终结符号 } c)$
将 $[B \rightarrow \cdot \gamma, c]$ 加入到集合 I 中

其他函数构造一样，不再赘述

LALR(1)

化简LR(1)：将具有相同核心的LR(1)项集，合并为一个项集。所谓项集和核心就是其第一分量的集合，即内核项

但将具有相同核心的状态替换为它们的并集，有可能产生冲突

见龙书p171