编译原理混子速成

闭包

$$A^0 = \{\epsilon\}$$
$$A^+ = \cup_{i=1}^{\infty} A^i$$
$$A^* = \cup_{i=0}^{\infty} A^i$$

文法 $G[S] = (V_N, V_T, P, S)$

正则表达式代数定律

定律	描述
$r\|s = s\|r$	|是可以交换的
$r\|(s\|t) = (r\|s)\|t$	|是可结合的
$r(st) = (rs)t$	连接是可结合的
$r(s\|t) = rs\|rt$ $(s\|t)r = sr\|tr$	连接对|是可分配的
$\epsilon r = r\epsilon = r$	ϵ是连接的单位元
$r^* = (r\|\epsilon)^*$	闭包一定包含ϵ
$r^{**} = r^*$	$*$具有等幂性

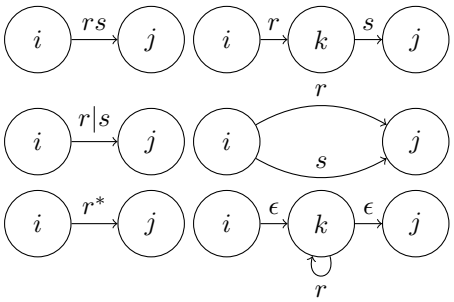
NFA 状态集上的操作

操作	描述
$\epsilon - closure(s)$	能够从NFA的状态s开始只通过ϵ转换到达的NFA状态的集合
$\epsilon - closure(T)$	能够从T中的某个NFA状态s开始只通过ϵ转换到达的NFA状态集合
$move(T, a)$	能够从T中的某个状态s出发通过标号为a的转换到达的NFA状态的集合

Algorithm 1: 子集构造法

$\epsilon - closure(s_0)$是$Dstates$中的唯一状态，且未加标记;
while 在$Dstates$中有一个未标记状态T **do**
  给T加上标记;
  for 每个输入符号 a **do**
    $U = \epsilon - closure(move(T, a))$;
    if U不在$Dstates$中 **then**
      将 U 加入到 $Dstates$ 中，且不加标记;
    end
    $Dtran[T, a] = U$;
  end
end

构造NFA



消除左递归

$$A \to A\alpha|\beta \Rightarrow \begin{cases} A \to \beta A' \\ A' \to \alpha A'|\epsilon \end{cases}$$

Algorithm 2: 消除左递归

按照某个顺序将非终结符号排序为 $A_1, A_2, \cdots, A_n$;
for $i \leftarrow 1$ *to* n **do**
  for $j \leftarrow 1$ *to* $i-1$ **do**
    $A_i \to A_j\gamma, A_j \to \delta_1|\delta_2|\cdots|\delta_n \Rightarrow A_i \to \delta_1\gamma|\delta_2\gamma|\cdots|\delta_k\gamma$;
  end
  消除A_i产生式之间的立即左递归;
end

提取左公因子

$$A \to \alpha\beta_1|\alpha\beta_2 \Rightarrow \begin{cases} A \to \alpha A' \\ A' \to \beta_1|\beta_2 \end{cases}$$

FIRST

- X 是终结符号，FIRST$(X) = X$。
- X 是非终结符号，

$$\begin{cases} X \to a\alpha, \text{则 } a \in \text{FIRST}(X) \\ X \to \epsilon, \text{则 } \epsilon \in \text{FIRST}(X) \end{cases}$$

- $X \to Y_1Y_2\cdots Y_k$
  1. Y_1 是非终结符，FIRST$(Y_1) - \{\epsilon\} \subseteq$FIRST$(X)$。
  2. $Y_1Y_2\cdots Y_i(1 \le i \le k-1)$均为非终结符且都能推出 ϵ，则FIRST(Y_1), FIRST(Y_2), $\cdots$, FIRST(Y_{i+1})中一切非ϵ $\in$FIRST(X)
  3. $Y_1, Y_2, \cdots, Y_k$都能推导出ϵ，则ϵ $\in$FIRST(X)

FOLLOW A为非终结符。

- 开始符 s，\$$\in$ FOLLOW(s)
- $B \to \alpha A\beta$

  FIRST$(\beta) - \{\epsilon\}$ $\in$FOLLOW(A)
-

$$\begin{cases} B \to \alpha A\beta \text{ 且 } \epsilon \in \text{FIRST}(\beta) \\ B \to \alpha A \end{cases}$$

  $\Rightarrow$ FOLLOW$(B) \in$ FOLLOW(A)

LL(1)

1. 已化简且无左递归。
2. 对 G 中每个产生式 $A \to \gamma_1|\gamma_2|\cdots|\gamma_m$，其各候选式均满足：
   (a) FIRST$(\gamma_i) \cap$FIRST$(\gamma_j) = \varnothing$
   (b) 若有候选式为ϵ，则其余候选式FIRST$(r_i) \cap$FOLLOW$(A) = \varnothing$

Algorithm 3: 预测分析表

对于FIRST(α)中的每个终结符号a，将$A \to a$加入到$M[A,a]$中;
如果ϵ在FIRST(α)中，那么对于FOLLOW(A) 中的每个终结符b，将$A \to \alpha$加入到 $M[A,b]$ 中。注意加入 \$ 符号;
其余设置为 error;

增广文法 加上新开始符号S'和产生式$S' \to S$。

Algorithm 4: CLOSURE(I)

```
J = I;
repeat
    for J中的每个项A → α · Bβ do
        for G的每个产生式B → γ do
            if 项B → ·γ不在J中 then
                将B → ·γ加入J中;
            end
        end
    end
until 在某一轮中没有新的项被加入到J中;
return J;
```

LR(0)项目

1. 归约项目($A \to \alpha\cdot$)
2. 接受项目($S \to \alpha\cdot$)
3. 移进项目($A \to \alpha \cdot x\beta$)
4. 待约项目($A \to \alpha \cdot X\beta$)

LR(0)分析表

ACTION si 遇到此终结符移到n状态

rj 此状态n为归约项目，此行都填rn

acc 接受项目，\$处填acc

GOTO 填n，经过此非终结符到达的状态n

SLR(1)

- DFA中存在冲突项目（移进—归约，归约—归约）
- $I_n = \{A_1 \to \alpha \cdot a_1\beta, A_2 \to \alpha \cdot a_2\beta \cdots$，$B_1 \to \alpha\cdot, B_2 \to \alpha \cdots\}$ 当$\{a_1, a_2, \cdots, a_m\}$，FOLLOW($B_1$)，FOLLOW($B_2$)$\cdots$两两不相交时为SLR(1) 文法。

Algorithm 5: SLR(1)分析表

```
构造 G' 的规范 LR(0) 项集族C = {I_0, I_1, ··· , I_n};
根据I_i构造得到状态i。 foreach 状态i do
    if [A → α · aβ]在I_i中并且GOTO(I_i, a) = I_j then
        ACTION[i, a] ←移入j;
    end
    if [S' ≠ A → α·]在I_i中 then
        FOLLOW(A)中所有的a，
        ACTION[i, a] ←规约A → α;
    end
    if S' → S·在I_i中 then
        ACTION[i, $] ←接受;
    end
end
foreach 状态i对于各个非终结符号A do
    if GOTO(I_i, A) = I_j then
        GOTO[i, A] = j;
    end
end
```

LR(1)项 $[A \to \alpha \cdot \beta, a]$

$A \to [\alpha \cdot B\beta, a]$ 其等价项目 $B \to [\cdot\gamma, b]$中，b =FIRST(βa)：$\beta = \epsilon$时，$b = a$继承；$\beta \neq \epsilon$时，b =FIRST(β)。

LALR(1) 合并同心集。

- 如果构造 LR(0) 的 DFA
  - 没有归约冲突就是 LR(0) 文法。
  - 有冲突但是可以通过 FOLLOW 集合解决冲突就是 SLR 文法。
  - 否则不是 SLR 文法。
- 如果构造 LR(1) 的 DFA
  - 没有冲突就是 LR(1) 文法。
  - 如果合并同心集之后也没有冲突，那么就是 LALR(1) 文法。

LR(0) < SLR < LALR < LR(1)

S属性 如果一个 SDD 的每个属性都是综合属性，它就是 SDD 的。

L属性 在一个产生式体的关联各个属性之间，依赖图的边总是从左到右，而不能从右到左。

四元式 $(op, arg_1, arg_2, result)$

Algorithm 6: 确定活跃性

```
foreach B的最后语句开始反向扫描 i : x = y + z do
    在符号表中找到的有关x, y, z的当前后续使用信息与语句i关联起来;
    在符号表中，设置x为“不活跃”和“无后续使用”;
    在符号表中，设置y与z为“活跃”，并把它们的下一次使用设置为语句i;
end
```

Algorithm 7: 寄存器分配

```
repeat
    foreach 节点n do
        if deg n < k then
            删除n及相连边;
        end
    end
until 没有更多的修改;
if 空图 then
    相反的删除顺序涂色;
else
    节点溢出后再次尝试;
end
```