编译原理

第二章 编译基础知识

方徽星

扬州大学 信息工程学院(505)

fanghuixing@yzu.edu.cn

2018年春季学期

本章主要内容

一．字母表、符号串、闭包

二．文法与语言的关系

三．文法构造

四．语法树与二义文法

第一节 字母表、符号串、闭包

第一节 字母表、符号串、闭包

- **字母表(Alphabet)**：非空有穷的符号**集合**
  - 典型的符号例子有字母、数字、标点符号
  - 集合{0，1}是二进制字母表(Binary Alphabet)
  - ASCII是字母表的一个重要例子，用于很多软件系统
  - Unicode包含了大约100,000个来自世界各地的字符，也是字母表的一个重要例子
- **符号串/串(String)**：由字母表中的符号组成的有穷序列
  - 术语“句子”和“字”常常作为“串”的同义词
  - 串s的长度，通常记作|s|，如：|abccba|=6
  - 空串(Empty String)是长度为0的串，用**ε**表示

第一节 字母表、符号串、闭包

- **串的各部分术语**
  - **前缀(Prefix)**：从串s的尾部删除0个或多个符号后得到的串，如：ban、banana和ε是banana的前缀
  - **后缀(Suffix)**：从串s的开始处删除0个或多个符号后得到的串，如：nana、banana和ε是banana的后缀
  - **子串**(Substring)：删除串s的某个前缀和某个后缀之后得到的串，如：banana、nan和ε是banana的字串
  - **真前缀、真后缀、真子串**：分别是串s的既不等于ε，也不等于s本身的前缀、后缀和子串
  - **子序列**(Subsequence)：从串s中删除0个或多个符号后得到的串，被删除的符号可能不相邻，如：baan是banana的一个子序列

第一节 字母表、符号串、闭包

- **语言(Language)**：某个给定字母表上一个任意的可数的串集合
  - 这个定义非常宽泛
  - 空集∅和仅包含空串的集合{ε}都是语言
  - 所有语法正确的c程序的集合
  - 所有语法正确的英语句子的集合
- **句子**：属于语言的串

1.1 符号串集合的运算

- 符号串集合的**乘积**(也称**联结**)
  - 串x，y的乘积记作xy，即把y附在后面而形成的串，如：x = dog，y = house，那么xy = doghouse
  - 空串ε是单位元，对于任何串s，都有sε = εs = s
  - 串s的指数运算(也称幂运算)
    - 定义s^0 为ε
    - 对于$i > 0$, 定义s^i为$s^{i-1}s$
  - 串集A与串集B的乘积$AB = \{st \mid s \in A, t \in B\}$
  - 串集A的幂： $A^0 = \{\varepsilon\}$，$A^n = A^{n-1}A$ $(n > 0)$

1.2 字母表的闭包和正闭包

- **字母表A的闭包**
  - $A^* = \bigcup_{i=0}^{\infty} A^i$
  - 由A中符号组成的所有串的集合，包括空串ε
- **字母表A的正闭包**
  - $A^+ = \bigcup_{i=1}^{\infty} A^i$
  - 不包括空串ε
- **语言的运算(对比)，设L、M均为语言**
  - 并：$L \cup \mathrm{M} = \{s \mid s \in L \ or \ s \in \mathrm{M}\}$
  - 联结：$LM = \{ st \mid s \in L \ and \ t \in \mathrm{M}\}$
  - 幂：L^0是$\{\varepsilon\}$，$L^n = L^{n-1}L \ (n > 0)$
  - 闭包：$L^* = \bigcup_{i=0}^{\infty} L^i$
  - 正闭包：$L^+ = \bigcup_{i=1}^{\infty} L^i$

第二节 文法与语言的关系

2.1 文法

- 文法G是一个四元组(V_N, V_T, P, S)，其中
  - V_N：非终结符号集合
  - V_T：终结符号集合
  - P：产生式集合，产生式格式：$\alpha \rightarrow \beta$
  - S：开始符号，是一个非终结符号
- 例子
  - 设文法G = ({S}, {a, b}, P, S)，其中产生式P定义如下：P = {S->aS, S->a, S->b}.
- 根据产生式的特点，文法可划分为4个层次

2.1 文法

- 0-型文法(无限制文法或短语文法)
- 产生式$\alpha \to \beta$满足如下条件约束
  - $\alpha, \beta \in (V_N \cup V_T)^*$
  - α中至少含有一个非终结符号
  - α不为空
- 例子
  - P ={S->ACaB, Bc->acB, CB->DB, aD->Db}.
- 能够产生所有可被图灵机识别的语言(指能够使图灵机停机)，又称为递归可枚举语言

2.1 文法

- 1−型文法(Context-sensitive grammar)
- 产生式$\alpha A\beta \rightarrow \alpha\gamma\beta$满足如下条件约束
  - $A \in V_N$
  - $\alpha, \beta, \gamma \in (V_N \cup V_T)^*$
  - α，β可能为空
  - γ不为空
  - 如果S不在任何产生式的右边，则**S → ε也可以**
- 例子
  - P ={AB−>AbBc， A−>bcA， B−>b}.
- 能够生成上下文相关语言,这种文法规定的语言可以被线性有界非确定图灵机接受

2.1 文法

- 2–型文法(Context-free Grammar)
- 产生式$A \to \gamma$满足如下条件约束
  - $A \in V_N$
  - $\gamma \in (V_N \cup V_T)^*$
- 例子
  - P ={S–>Xa，X–>a，X–>aX，X–>abc，X–> ε }.
- 生成上下文无关语言，可被非确定下推自动机接受

2.1 文法

- 3–型文法(Regular Grammar)
  - 右线性文法产生式：$X \to \alpha$或者$X \to \alpha Y$
  - 左线性文法产生式：$X \to \alpha$或者$X \to Y\alpha$
- 约束条件：
  - $X, Y \in V_N$
  - $\alpha \in V_T^*$
- 例子
  - P ={X–> ε，X –>（a | aY)，Y–>b}.
- 生成正规语言，可被有限状态自动机接受，也可以通过正则表达式来获得

2.1 文法

- 关系
  - 从0-型到1-型，限制条件逐渐变强
  - 1～3-型文法都属于0-型文法
- 区别
  - 0-型和1-型文法产生式左部存在含有终结符号的符号串或两个以上的非终结符
  - 2-型和3-型文法的产生式左部只可以是单个非终结符

2.1 文法

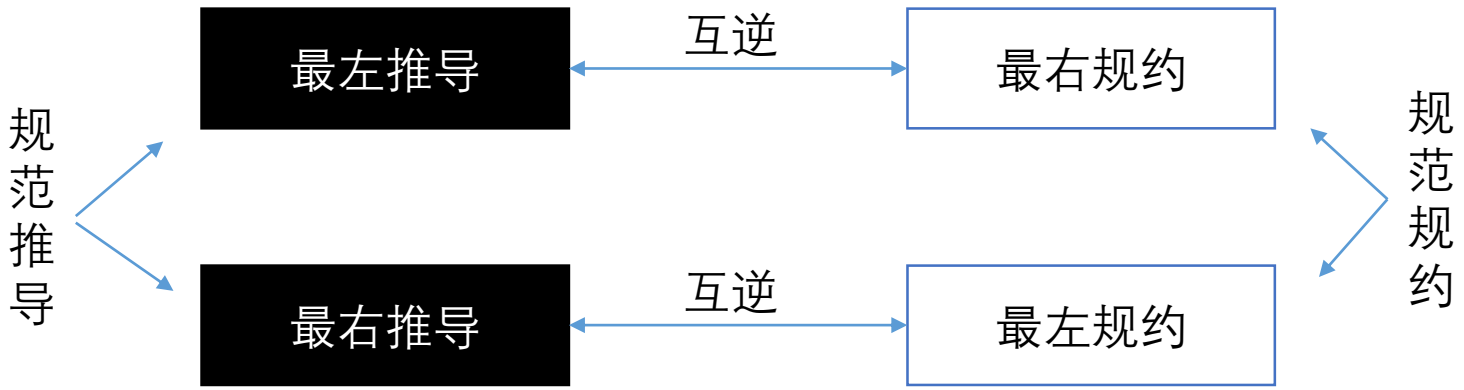
- 文法的BNF(Backus-Naur Form)表示法规定的推导规则(即产生式)：

  〈符号〉::=〈表达式〉

  - 这里的〈符号〉是非终结符，而表达式由一个符号序列，或用竖杠'|'分隔的多个符号序列构成
  - 每个符号序列整体都是左端的符号的一种可能的替代
  - 从未在左端出现的符号叫做终结符
- EBNF增加：
  - []：任选符号，[]内符号至多出现一次
  - ()：提因子符号，候选式中有公共因子时可外提
  - {}：重复符号
- 元语言(Meta Language)符号：用来说明文法符号之间的关系，如：‘->’，‘|’

2.2 语言

- **推导与规约**
  - 推导：从文法的开始符号开始，通过产生式的右部取代左部的过程，最终能产生一个语言的句子
  - 规约：从给定源语言的句子开始，通过产生式的左部取代右部的过程，最终到达开始符号



2.2 语言

- **最左推导**：每次使用一个产生式以其右部取代符号串的**最左**非终结符
- **最右推导**：每次使用一个产生式以其右部取代符号串的**最右**非终结符
- 例子，设有产生式E -> E + E | E * E | (E) | - E | id，则有
  - 最左推导：

  E => -E => -(E) => -(E + E)=> -(id + E) => -(id + id)

  - 最右推导：

  E => -E => -(E) => -(E + E) => -(E + id) => -(id + id)

2.2 语言

- **句型**：假设G是一个文法，S是开始符号，则从S开始每步推导（包含0步）所得字符串α称作句型，记为

$$S \overset{*}{\rightarrow} \alpha, \text{其中} \alpha \in (V_N \cup V_T)^*$$

- **句子**：仅含终结符的句型
- **语言**：由S开始通过1步以上推导所得的句子所组成的集合，记为

$$L(G) = \{\alpha \mid S \overset{+}{\rightarrow} \alpha, \alpha \in V_T^*\}$$

第三节 文法构造

3.1 由语言如何构造文法

- 设语言$L_1 = \{a^{2n}b^n | n \geq 1$且$a, b \in V_T\}$，试构造生成$L_1$的文法$G_1$.

  解：

n	句子	产生式
1	aab	S->aab
2	aaaabb	S->aa***aab***b => S->aaSb
3	aaaaaabbb	S->aa**aaaabb**b => S->aaSb

  G_1的产生式P：

  1. S -> aab
  2. S -> aaSb

3.1 由语言如何构造文法

- 设语言$L_2 = \{a^i b^j c^k | i,j,k \geq 1$且$a,b,c \in V_T\}$，试构造生成$L_2$的文法$G_2$.

  解：

  - 这是由a, b, c字母组成的串；
  - 串中a, b, c的数目分别大于1；
  - 串中a排在前面，b居中，c为串尾；
  - G_2的产生式P：

    1. S -> aS | aB
    2. B -> bB | bA
    3. A -> cA | c

3.1 由语言如何构造文法

- 设$L_3 = \{\omega | \ \omega \in \{a, b\}^*$, 且$\omega$中$a$和$b$个数相同$\}$
  试构造L_3的文法G_3.

  解:

  - 由于L_3允许包含空串，所以有产生式：S-> ε ;
  - 串ω可以是a开头，也可以是b开头，所以有产生式：S -> aA，S -> bB;
  - 串ω中a和n的个数相同，则A产生的串满足约束：b比a多1个，因此有A -> bS; A -> aAA;
  - 同理有产生式：B -> aS，B -> bBB.

3.1 由语言如何构造文法

- 设$L_4 = \{\omega | \ \omega \in \{0,1\}^*$,且$\omega$中1的个数为偶数$\}$
  试构造L_4的文法G_4.

  解：

  - 由于L_4允许包含空串，所以有产生式：S -> ε ;
  - 串ω可以是0开头的,所以有：S -> 0S;
  - 串ω可以是1开头的,但后续需要有额外一个1与之对应从而使得1的个数满足偶数约束：S -> 1A;
  - 非终结符A可以从0开头，也可以从1开头
    - 0开头时，不限制0的个数：A -> 0A;
    - 1开头时，后续1的个数需为奇数：A -> 1S.

3.2 由正规式如何构造文法

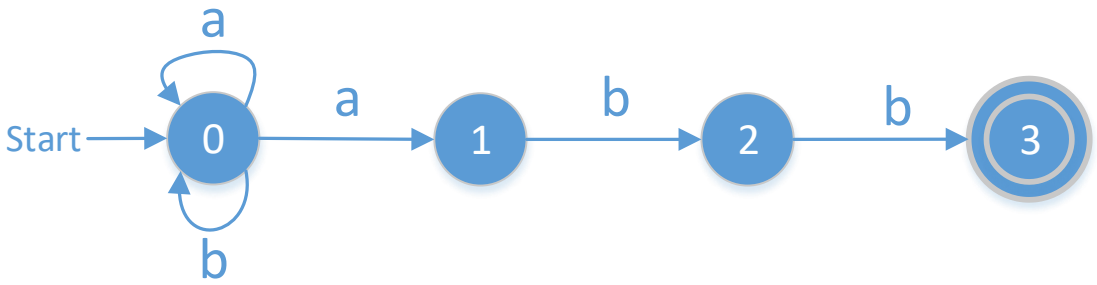
- 正规式/正则表达式(Regular Expression)
  - 空串符号 ε 和空集符号∅均是正规式;
  - 如果符号$a \in \Sigma$，则a是一个正规式;
  - 如果α, β是正规式，则下面都是正规式:
    - $\alpha \mid \beta$ （或）
    - $\alpha \cdot \beta$（简写为$\alpha\beta$） （连接）
    - α^*和 β^* （闭包）
- 优先级: $* > \cdot > \mid$
- 常使用成对圆括号组织正规式

3.2 由正规式如何构造文法

- 设正规式$R = (a|b)^*abb$，构造其文法G.

  解：

  - 首先把正规式R转成非确定有限自动机(NFA)：



  - 其次按照如下步骤构造文法G：
    - 对于NFA的每个状态i，创建一个非终结符号A_i；
    - 如果$i \xrightarrow{a} j$，则新增产生式$A_i \to aA_j$；
    - 如果$i \xrightarrow{\varepsilon} j$，则新增产生式$A_i \to A_j$；
    - 如果i是一个接受状态，则新增产生式$A_i \to \varepsilon$；
    - 如果i是开始状态，令A_i为所得文法的开始符号.

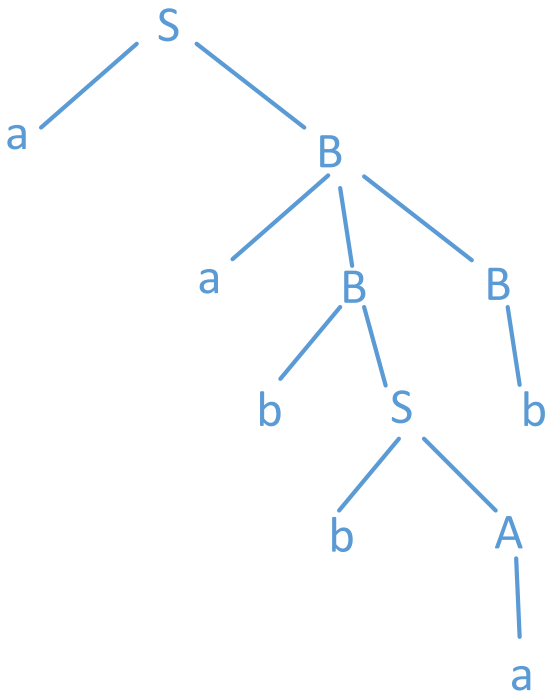
第四节 语法树与二义文法

4.1 语法树

- 语法树是文法**推导**的树形**表示形式**，且过滤掉了推导过程中对非终结符号应用产生式的顺序
- 语法树的**内部**结点表示一个产生式的应用，内部结点使用产生式头中的**非终结符**标记
  - 内部结点的子结点的标号从左自右组成了在推导过程中替换非终结符的产生式体
- 每个**叶子**结点的标号为一个终结符号或者 ε

4.1 语法树

- 设有一个2-型文法G=({S,A,B}, {a,b}, P, S),其中P:
  1. S->aB
  2. S->bA
  3. A->a
  4. A->aS
  5. A->bAA
  6. B->aBB
  7. B->bS
  8. B->b



产生句子aabbab的语法树

4.2 文法的二义性

- 如果一个文法可以为某个句子生成多棵语法树，则该文法是二义性的(ambiguous)
- 二义性文法就是对同一个句子有多个最左(右)推导的文法
- 例子
  - 文法： E -> E+E | E×E | (E) | id
  - 推导：句子id+id×id

1. E=>E+E
=>id+E
=>id+E×E
=>id+id×E
=>id+id×id

2. E=>E×E
=>E+ E×E
=>id+E×E
=>id+id×E
=>id+id×id

4.2 文法的二义性

- 对于文法：E -> E+E | E×E | (E) | id，可以找到一个非二义性文法：
  - E->T|E+T
  - T->F|T×F
  - F->(E) | id
- 如何消除二义性
  - 增加限制条件
  - 组织优先级
- 文法二义性问题：不存在一种算法可以在有限时间内判定一个文法是否是二义的

小结

- 概念：字母表、符号串、闭包
- 形式定义：文法、语言
- 从语言、正规式构造文法
- 介绍：语法树、文法二义性

图灵机

- 图灵的基本思想是用机器来模拟人们用纸笔进行数学运算的过程，他把这样的过程看作下列两种简单的动作：
  - 在纸上写上或擦除某个符号；
  - 把注意力从纸的一个位置移动到另一个位置；
- 而在每个阶段，人要决定下一步的动作，依赖于
  - 此人当前所关注的纸上某个位置的符号和
  - 此人当前思维的状态

图灵机

- 为了模拟人的这种运算过程，图灵构造出一台假想的机器：
  - 一条无限长的纸带TAPE。纸带被划分为一个接一个的小格子，每个格子上包含一个来自有限字母表的符号，字母表中有一个特殊的符号□表示空白。纸带上的格子从左到右依次被编号为0，1，2，...，纸带的右端可以无限伸展
  - 一个读写头HEAD:可以在纸带上左右移动，它能读出当前所指的格子上的符号，并能改变当前格子上的符号

图灵机

- 为了模拟人的这种运算过程，图灵构造出一台假想的机器：
  - 一套控制规则TABLE:根据当前机器所处的状态以及当前读写头所指的格子上的符号来确定读写头下一步的动作，并改变状态寄存器的值，令机器进入一个新的状态，按照以下顺序告知图灵机命令：
    - 写入（替换）或擦除当前符号;
    - 移动 HEAD， 'L'向左， 'R'向右或者'N'不移动;
    - 保持当前状态或者转到另一状态
  - 一个状态寄存器：用来保存图灵机当前所处的状态
  - 图灵机的所有可能状态的数目是有限的，并且有一个特殊的状态，称为停机状态