编译原理 September 28, 2011

Homework 2

计算机 92 班 戴唯思 09055029

1

试论词法分析器的自动生成器的理论基础

词法分析是指将输入的字符序列转换为单词 (token) 序列的过程, 其中单词是构成源代码的最小单位. 词法分析器的设计需要根据特定语言的词法规则, 词法规则可以用正规式表示.

词法分析器的自动生成器的理论基础是控制规则的普遍性 与正规式和 DFA 的等价性.

词法分析器主要由控制逻辑和转化表构成. 由同一个词法分析器生成器生成的不同的词法分析器的控制逻辑是一样的, 它的作用是从输入流读入字符 → 在 DFA 上运行 → 执行最大串匹配 → 判别是否达到终态 → 构造二元组返回, 在满足上下文无关文法的前提下, **一套控制逻辑适用于一切语言**.

除此之外, 词法分析器还要生成转化表, 而**转化表可以根据词法规则经过机械操作生成得到** (将 NFA 合并, 确定化得到等价的 DFA). 例如, Lex/flex 的源程序包括了正规定义式和识别规则两部分.

2

阐述从文法定义语言过程及各概念的关系

形式语言是指在某个字母表上, 一些有限长字串的集合, 形式文法是描述形式语言的一种方法.

文法 (grammar) 是描述语言的语法结构的形式规则, 目的是解决语言的有穷说明问题, 包含对语法的描述, 但却不表达任何语义. 文法的描述应当: 形式上严格准确, 易于理解, 具有较强的描述能力, 有利于句子的分析和翻译, 构造语法分析器.

形式文法的基本思想是, 从初始符号出发, 应用产生式规则得到一系列字串的集合.

文法分为 4 类, 对应 4 类语言. 与程序语言语法有关的是上下文无关文法 (2 型文法).

上下文无关文法定义的语法范畴 (或语法单位) 完全独立于这种范畴可能出现的环境. 它只能描述一部分语言. 上下文无关文法 G 是一个四元式 $(V_T, V_N, S, \mathcal{P})$, 其中 V_T 是非空有限集, 元素是终结符号, V_N 元素是非终结符号, $S \in V_N$ 称为开始符号, $\mathcal{P}$ 是产生式集合, 每个产生式形式是 $\{P \to \alpha | P \in V_N, \alpha \in (V_T \cup V_N)^*, \mathrm{S}$ 至少一次为 $\mathrm{P}\}$. 其中,

终结符号 用以组成语言中的串的**基本**符号;

非终结符 是标记某种**串**的集合的特定符号;

开始符号 一个非终结符号, 标记感兴趣的语法范畴, 定义文法的顶层.

产生式规定由终结符和别的语法范畴组成一个新的语法范畴的方法, 语法结构: 非终结符 $\to$ 一串非终结符和终结符.

为了用有穷条产生式得到无穷的表达式语言, 必须采用递归.

直接推导是两个字符串之间的一种关系: $(\alpha A\beta)\mathcal{R}(\alpha\gamma\beta)$, 若 $A \to \gamma \in \mathcal{P}, \alpha, \beta \in V^*, V = V_T \cup V_N$, 则 $\mathcal{R}$ 就是直接推导, 记为 $\alpha A\beta \Rightarrow \alpha\gamma\beta$.

推导是两个串 u_0, u_n 存在一个串序列 $u_0 \Rightarrow u_1 \Rightarrow \cdots \Rightarrow u_n$, 则 $u_0\mathcal{R}_1u_n$. $u_0 \overset{+}{\Rightarrow} u_n$ 表示从 u_0 出发, 经过一步或若干步可推导出 u_n. $u_0 \overset{*}{\Rightarrow} u_n$ 表示从 u_0 出发, 经过零步或若干步可推导出 u_n.

要由推导引出语言, 限制推导中的 u_0 为 S, 推导要从开始符号开始. $S \overset{*}{\Rightarrow} \alpha, \alpha \in V^*$, 称 α 为 G 的句型, 如再要求 $\alpha \in V_T^*$, 则 α 为 G 的句子. 文法 G 所产生的句子的全体是一个语言, 记为 $L(G) = \{\alpha | S \overset{+}{\Rightarrow} \alpha \text{ 且 } \alpha \in V_T^*\}$. 从一个句型到另一个句型的推导过程一般不唯一, 通常只考虑最左推导和最右推导.

3

文法如下:

1. $E \to TE'$
2. $E' \to +TE'|\varepsilon$
3. $T \to FT'$
4. $T' \to *FT'|\varepsilon$
5. $F \to (E)|i$

对 $i_1i_4 + i_2 * i_3$ 和 $i_1 + i_2 * i_3i_5$ 分析器如何动作, 是何结果?

1. $i_1i_4 + i_2 * i_3$

栈	输入	输出
$\#E$	$\underline{i_1}i_4 + i_2 * i_3$	
$\#E'T$	$\underline{i_1}i_4 + i_2 * i_3$	$E \to TE'$
$\#E'T'F$	$\underline{i_1}i_4 + i_2 * i_3$	$T \to T'F$
$\#E'T'i$	$\underline{i_1}i_4 + i_2 * i_3$	$F \to i$
$\#E'T'$	$\underline{i_4} + i_2 * i_3$	
$\#E'\underline{T'}$	$\underline{i_4} + i_2 * i_3$	error

匹配失败.

2. $i_1 + i_2 * i_3i_5$

栈	输入	输出
$\#E$	$\underline{i_1} + i_2 * i_3 i_5$	
$\#E'T$	$\underline{i_1} + i_2 * i_3 i_5$	$E \to TE'$
$\#E'T'F$	$\underline{i_1} + i_2 * i_3 i_5$	$T \to FT'$
$\#E'T'$	$\underline{+} i_2 * i_3 i_5$	$F \to i_1$
$\#E'$	$\underline{+} i_2 * i_3 i_5$	$T' \to \varepsilon$
$\#E'T+$	$\underline{+} i_2 * i_3 i_5$	$E' \to +TE'$
$\#E'T$	$\underline{i_2} * i_3 i_5$	匹配 $+$
$\#E'T'F$	$\underline{i_2} * i_3 i_5$	$T \to FT'$
$\#E'T'$	$\underline{*} i_3 i_5$	$F \to i_2$
$\#E'T'F*$	$\underline{*} i_3 i_5$	$T' \to *FT$
$\#E'T'F$	$\underline{i_3} i_5$	匹配 $*$
$\#E'T'$	$\underline{i_5}$	$F \to i_3$
$\#E'$	$\underline{i_5}$	error

匹配失败.

References

1. https://secure.wikimedia.org/wikipedia/en/wiki/Lexical_analysis#Lexer_generator
2. http://zh.wikipedia.org/zh/%E5%BD%A2%E5%BC%8F%E6%96%87%E6%B3%95