Compile Principle: A Simple SDT **Spring 2025**

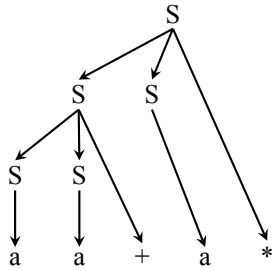
Homework 2 — March 6

Lecturer: Feng Xiaobing *Completed by: 2022K8009929010 Zhang Jiawei*

2.1

(1) S $\xrightarrow{S\to SS*}$ SS* $\xrightarrow{S\to SS+}$ SS+S* $\xrightarrow{S\to a}$ aS+S* $\xrightarrow{S\to a}$ aa+S* $\xrightarrow{S\to a}$ aa+a*

(2) 构建的语法分析树如下：



(3) 该文法生成的语言是后缀表达式（逆波兰表示法），其中终结符为字母 a 和运算符 $+$、$*$，形式化定义为：$L = \{w \mid w$ 是由$a, +, *$ 组成的合法后缀表达式$\}$

从产生式 $S \to a \mid SS+ \mid SS*$ 可以看出，每个 $SS+$ 或 $SS*$ 结构都表示一个完整的后缀表达式，其中运算符位于其两个操作数之后。

(4) 该文法不具有二义性。对于任意一个合法的后缀表达式，只存在唯一一棵语法分析树。这是因为我们可以通过从左到右读取表达式，使用栈数据结构来确定唯一的语法结构。

当读取到操作符时，它必定对应于之前最近的两个完整表达式（操作数），这种关系是确定的。例如，对于表达式"ab+"，操作符"+" 唯一地确定了它操作的是前面的"a" 和"b"。

因此，对于任何合法的后缀表达式，我们总能构造出唯一的推导序列和语法分析树，证明该文法没有二义性。

2.2

(1) 证明过程如下：

证明 记 $val(num)$ 为非终结符 num 推导出的二进制串所表示的十进制值。我们将对语法分析树的节点数进行数学归纳法来证明 $val(num) \equiv 0 \pmod 3$。

考虑具有 2 个节点的最小语法分析树（一个节点是 num，一个节点是终结符串）：

- 对于 $num \to 11$，$val(num) = 3$，能被 3 整除。
- 对于 $num \to 1001$，$val(num) = 9$，能被 3 整除。

假设对于所有节点数不超过 n 的语法分析树($n \geq 2$),其对应的二进制串的值都能被 3 整除。

考虑有 $n+1$ 个节点的语法分析树。根节点的左右子树必须通过以下两种产生式之一得到:

情况 1:$num \to num\ 0$ 在这种情况下,左边 num 的子树最多有 n 个节点。根据归纳假设,这个子树的值 $val(num') \equiv 0 \pmod 3$。因此 $val(num) = 2 \cdot val(num') \equiv 2 \cdot 0 \equiv 0 \pmod 3$,即 $val(num)$ 能被 3 整除。

情况 2:$num \to num\ num$ 在这种情况下,左右两个 num 的子树都少于 $n+1$ 个节点。设它们的值分别是 $val(num_1)$ 和 $val(num_2)$。根据归纳假设,$val(num_1) \equiv 0 \pmod 3$ 且 $val(num_2) \equiv 0 \pmod 3$。

整个字符串的值为 $val(num) = val(num_1) \cdot 2^{l_2} + val(num_2)$,其中 l_2 是右边 num 推导出的二进制串的长度。由于 $val(num_1) \equiv 0 \pmod 3$ 且 $val(num_2) \equiv 0 \pmod 3$,我们有:$val(num) = val(num_1) \cdot 2^{l_2} + val(num_2) \equiv 0 \cdot 2^{l_2} + 0 \equiv 0 \pmod 3$

因此,由 $n+1$ 个节点的语法分析树表示的二进制串也表示能被 3 整除的数。根据数学归纳法原理,我们证明了该文法生成的所有二进制串的值都能被 3 整除。 □

(2) 所有能被 3 整除的数不一定可以被这个文法生成。

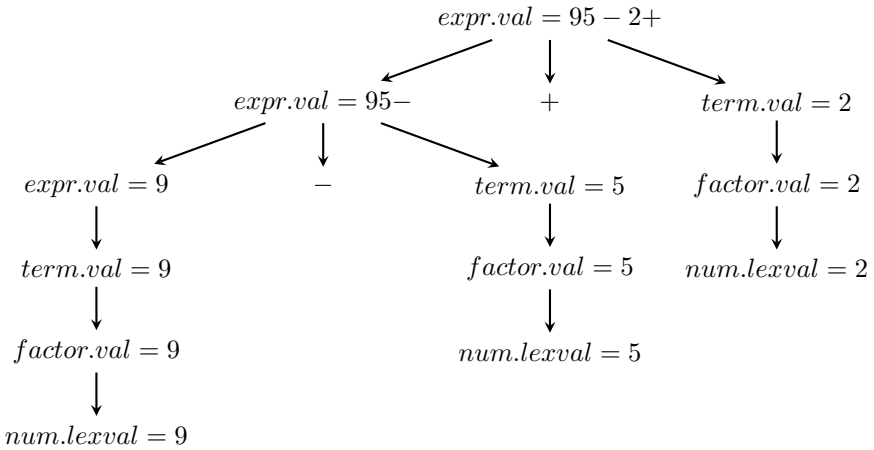
证明 考虑 21 这个数,它能被 3 整除,但是无法通过这个文法生成。因为 21 的二进制表示是 10101,没有连续的两个 1 和连续的两个 0,而两个终结符串为 11 和 1001,故无法生成 10101。 □

2.3

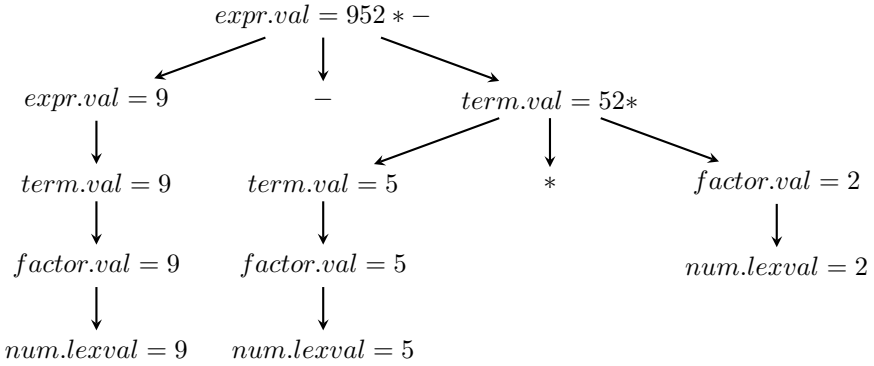
所构建的语法制导翻译方案如下:

$$
\begin{aligned}
expr &\to expr + term\{print('+')\} \\
expr &\to expr - term\{print('-')\} \\
expr &\to term \\
term &\to term * factor\{print('*')\} \\
term &\to term/factor\{print('/')\} \\
term &\to factor \\
factor &\to (expr) \\
factor &\to num\{print(num)\}
\end{aligned}
$$

对于中缀表达式 $9-5+2$,使用所给语法制导翻译方案构建的注释分析树如下:



对于中缀表达式 $9-5*2$,使用所给语法制导翻译方案构建的注释分析树如下:



2.4

在 C 语言中:

- for 语句开始时,创建一个新的作用域用于初始化子句
- 处理初始化子句中的声明并将标识符添加到这个新作用域
- 处理条件表达式
- 处理循环体之前,再创建一个新的嵌套作用域
- 处理循环体中的语句,将新声明的标识符添加到循环体作用域
- 处理更新表达式
- 循环体执行完后,退出循环体作用域
- for 语句结束后,退出初始化子句作用域

在 C++ 中:

- for 语句开始时,创建一个新的作用域
- 处理初始化子句中的声明并将标识符添加到此作用域
- 处理条件表达式
- 处理循环体中的语句,将新声明的标识符添加到同一个作用域(不允许重名)
- 处理更新表达式
- for 语句结束后,退出此作用域

2.5

所设计的文法如下:

$$
\begin{aligned}
S &\rightarrow 1 \\
S &\rightarrow 1D1 \\
S &\rightarrow 10D1 \\
D &\rightarrow 0 \\
D &\rightarrow 1 \\
D &\rightarrow 0D \\
D &\rightarrow 1D
\end{aligned}
$$