

3 上下文无关文法

徐辉, xuh@fudan.edu.cn

本章学习目标:

- 掌握上下文无关文法
- 掌握上下文无关文法的二义性问题和消除方法
- 使用扩展 BNF 范式定义上下文无关语言

3.1 上下文无关文法

定义 1 (上下文无关文法 (CFG: Context-free Grammar)). 由一系列形如 $X \mapsto \gamma$ 的规则或产生式组成, 其中 X 是一个非终结符, γ 是由终结符 (terminal symbol) 或非终结符组成的字符串。

注意, CFG 文法每一条规则的左侧只有一个非终结符, 不含其它限制条件, 否则不是 CFG 文法。例如, $aX \mapsto ab$, $bX \mapsto bc$ 的左侧对于如何展开 X 有其它前置条件限制。

$$\begin{aligned} E &\mapsto E \text{ '+' } E \\ E &\mapsto E \text{ '-' } E \\ E &\mapsto E \text{ '*' } E \\ E &\mapsto E \text{ '/' } E \\ E &\mapsto E \text{ '^' } E \\ E &\mapsto \text{'(' } E \text{ ')'} \\ E &\mapsto \text{NUM} \\ \text{NUM} &\mapsto \text{'<UNUM>} \\ \text{NUM} &\mapsto \text{'-' } \text{'<UNUM>} \end{aligned} \tag{3.1}$$

语法 3.1 尝试用 CFG 文法定义计算器算式输入, 即从 E 开始应用其中的语法规则可以得到所有合法的算式。

3.2 二义性问题和消除

语法 3.1 定义的文法对于特定算式可能会存在两种以上的推导方式, 带来二义性问题。以算式 $1+2*3$ 为例, 图 3.1 展示了两种可能存在的语法解析树。这两棵树对应的计算结果不同, 但只有解析树 1 是正确的。该二义性的原因是没有考虑运算符优先级。

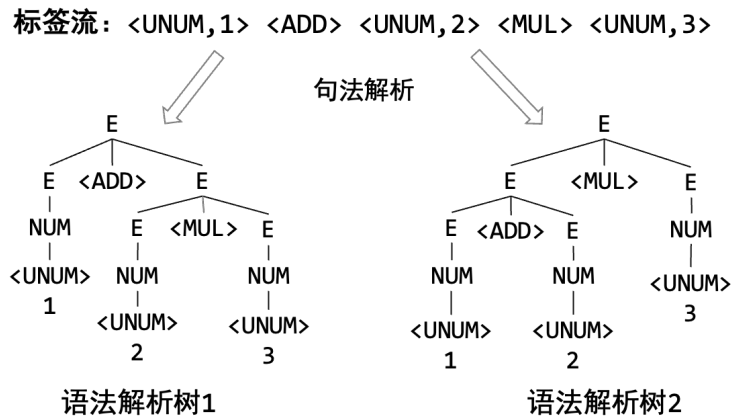


图 3.1: 语法解析: 算式 $1+2*3$

$$\begin{aligned}
 E &\mapsto E \text{ OP1 } E1 \\
 E &\mapsto E1 \\
 E1 &\mapsto E1 \text{ OP2 } E2 \\
 E1 &\mapsto E2 \\
 E2 &\mapsto E3 \text{ OP3 } E2 \\
 E2 &\mapsto E3 \\
 E3 &\mapsto \text{NUM} \\
 E3 &\mapsto ' (' E ') ' \\
 E &\mapsto \text{NUM} \\
 \text{NUM} &\mapsto \langle \text{UNUM} \rangle \\
 \text{NUM} &\mapsto ' - ' \langle \text{UNUM} \rangle \\
 \text{OP1} &\mapsto ' + ' \\
 \text{OP1} &\mapsto ' - ' \\
 \text{OP2} &\mapsto ' * ' \\
 \text{OP2} &\mapsto ' / ' \\
 \text{OP3} &\mapsto ' ^ '
 \end{aligned} \tag{3.2}$$

为了消除二义性，算式 3.2对算式 3.1进行了改写，将优先级和结合性信息加入语法规则中。主要思路是：

- 将运算符按照优先级分类并使用不同的符号表示。例如 OP1 表示优先级最低的加减运算， OP2 表示乘除运算， OP3 表示优先级最高的指数运算。
- 使用不同的符号表示不同优先级算子对应的运算数和运算结果。例如使用 E 表示加减运算的运算结果，它的运算数可以是优先级更高的乘除运算的结果 $E1$ ，也可以是同等优先级的加减运算结果 E ；为了保证文法等价性， E 也可以直接是乘除运算结果 $E1$ 。
- 使语法规则满足结合性。例如运算符 OP1 是左结合的，则其规则应为左递归形式 ($E \mapsto E \text{ OP1 } E1$)；如运算符 OP3 是右结合的，则其规则应为右递归形式 ($E2 \mapsto E3 \text{ OP3 } E2$)。

3.3 扩展 BNF 范式

由于 CFG 文法写起来比较复杂, 我们一般使用扩展 BNF 范式 (EBNF: Extended Backus-Naur Form) 来描述具体的上下文无关语言规则。表 3.1 给出了本文用到的 EBNF 构造符号。该符号参考了 PEG 文法中的构造符号表示方法 [1], 主要目的是与上节课学习的正则表达式构造符号兼容。语法 3.3 使用 EBNF 对语法 3.2 进行了改写。

表 3.1: 扩展 BNF 范式: 本文采用的文法构造符号。

构造方式	符号	示例	含义
字符匹配	' '	'ab'	匹配字符串 'ab'
任意字符	.	.	匹配任意字符
字符范围	[]	[a-z]	匹配任意 a-z 之间的字符
可选匹配	?	$\alpha?$	匹配 α 或 ϵ
非	!	$!\alpha$	匹配 α 之外的任意符号
连接		$\alpha\beta$	连续匹配 α 和 β
选择		$\alpha \beta$	匹配 α 或 β
闭包	*	α^*	匹配若干个 (≥ 0) 连续 α
正闭包	+	α^+	匹配若干个 (≥ 1) 连续 α

$$\begin{aligned} E &\mapsto \text{Factor } (('+' | '-') \text{Factor})^* \\ \text{Factor} &\mapsto \text{Power } (('*' | '/') \text{Power})^* \\ \text{Power} &\mapsto \text{Value } ('^' \text{Power})? \\ \text{Value} &\mapsto \langle \text{UNUM} \rangle \mid '-' \langle \text{UNUM} \rangle \mid '(' E ') ' \end{aligned} \tag{3.3}$$

练习

1. 以开发正则表达式工具（输入任意的正则表达式，生成对应的正则匹配器）为目标，设计用于解析正则表达式的 CFG 文法。
2. 使用 EBNF 对上述语法进行改写。

Bibliography

- [1] Bryan Ford. "Parsing expression grammars: a recognition-based syntactic foundation." In Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 111-122. 2004.