

## 5 自底向上解析

徐辉, xuh@fudan.edu.cn

本章学习目标:

- 了解自底向上解析思路
- 掌握 SLR 文法和解析方法
- 了解 LR(1)、LALR、GLR 解析方法

### 5.1 自底向上解析

自底而上解析指的是从句子开始, 逐步将其规约为语法规则初始符号的方法。本章主要讲解 LR(left-to-right, right-most) 的自底向上解析方法, 该方法包括包括两种基本操作:

- 移进 (shift): 读入句子中的下一个标签到解析栈
- 规约 (reduce): 根据语法规则  $X \mapsto \beta$  将当前解析栈顶的  $\beta$  规约为  $X$

该问题的难点在于实际解析过程中可能同时存在多种操作可能, 需要选择恰当的操作才能正确解析。本章以 SLR 文法为主详细讲解 LR 解析方法, 并在此基础上进一步探讨更多的扩展方法。

### 5.2 SLR 文法和解析

SLR 文法是一种特殊的 CFG 文法, 要求构建的 SLR 解析表不存在冲突。对于一套 SLR 文法, 其解析方法构造包括两个步骤: 1) 构造 LR(0) 自动机; 2) 创建 SLR 解析表。下面以计算器文法为例讲解 SLR 解析方法。

### 5.2.1 构造 LR(0) 自动机

由于计算器语法规则中初始符号对应的规则不唯一，为便于分析，我们在原始文法基础上增加一条目标文法  $S \mapsto E$ 。更新后的文法规则如文法 5.1所示。

$$\begin{aligned} S &\mapsto E \\ E &\mapsto E \text{ OP1 } E1 \\ E &\mapsto E1 \\ E1 &\mapsto E1 \text{ OP2 } E2 \\ E1 &\mapsto E2 \\ E2 &\mapsto E3 \text{ OP3 } E2 \\ E2 &\mapsto E3 \\ E3 &\mapsto \text{NUM} \\ E3 &\mapsto '(' E ')' \\ \text{NUM} &\mapsto <\text{UNUM}> \\ \text{NUM} &\mapsto '-' <\text{UNUM}> \\ \text{OP1} &\mapsto '+' \\ \text{OP1} &\mapsto '-' \\ \text{OP2} &\mapsto '*' \\ \text{OP2} &\mapsto '/' \\ \text{OP3} &\mapsto '^' \end{aligned} \tag{5.1}$$

从规范项  $S \mapsto \circ E$  开始，应用规范族生成算法 1 计算  $\text{RegularSet}(S \mapsto \circ E)$ ，我们可以得到一个规范族 5.2，即 LR(0) 自动机的初始状态  $S_0$ 。

---

#### 算法 1 规范族生成算法

---

```
procedure REGULARSET(Q)
  hasChanged  $\leftarrow$  TRUE
  while hasChanged do
    hasChanged  $\leftarrow$  FALSE
    for each  $A \mapsto \beta \circ C \delta \in Q$  do
      for each  $C \mapsto \lambda \in G$  do
        if  $C \mapsto \circ \lambda \notin Q$  then
           $Q \leftarrow Q \cup \{C \mapsto \circ \lambda\}$ 
          hasChanged  $\leftarrow$  TRUE
        end if
      end for
    end for
  end while
end procedure
```

---

$$\begin{aligned}
S &\mapsto \circ E \\
E &\mapsto \circ E \text{ OP1 } E1 \\
E &\mapsto \circ E1 \\
E1 &\mapsto \circ E1 \text{ OP2 } E2 \\
E1 &\mapsto \circ E2 \\
E2 &\mapsto \circ E3 \text{ OP3 } E2 \\
E2 &\mapsto \circ E3 \\
E3 &\mapsto \circ \text{NUM} \\
E3 &\mapsto \circ '(' E ') ' \\
\text{NUM} &\mapsto \circ \langle \text{UNUM} \rangle \\
\text{NUM} &\mapsto \circ '-' \langle \text{UNUM} \rangle
\end{aligned}
\tag{5.2}$$

LR(0) 自动机指的是规范族之间的状态转移关系。其构造方式是从  $S_0$  开始，分析其读入每个符号后得到的新的规范族，即新的状态；迭代该过程直至没有新的规范族和状态转移关系产生为止。图 5 展示了文法 5.1 对应的 LR(0) 自动机。

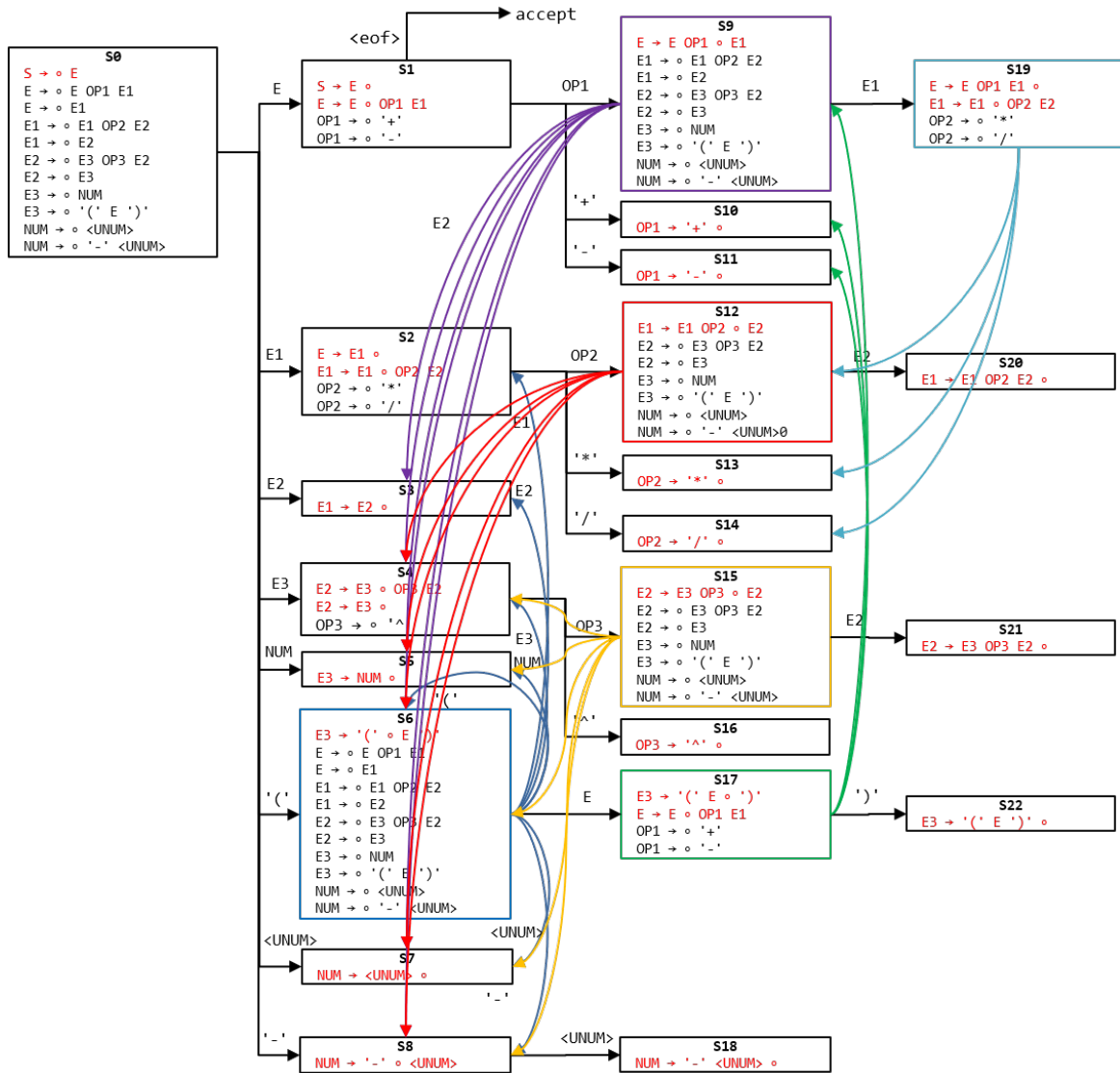


图 5.1: 计算器文法对应的 LR(0) 自动机

## 5.2.2 创建 SLR 解析表

根据 LR(0) 自动机便可以构造 SLR 解析表。如表 5.1 所示，每一行表示一个 LR(0) 自动机状态，每一列表示一个语法规则符号，每一个单元格表示 LR(0) 自动机读入特定符号后的目标状态。另外，还有一些单元格表示可应用的规约规则，如 R[2] 表示可将规范族中的规范项  $E \mapsto E1 \circ$  规约为 E。该规约并非在所有情况下都是对的，应满足的前提条件是下一个待读入的终结符属于  $Follow(E)$ 。因此，每个状态可应用的规约操作只出现在 SLR 解析表特定的列中。一般根据语法规则符号是否为终结符将 SLR 表分为左右两部分：Goto（对应非终结符）和 Action（对应终结符）。仅 SLR 表的 Action 部分有规约操作。

表 5.1: 计算器文法对应的 SLR 解析表

规范族	Goto								Action (Shift-Reduce)								
	E	E1	E2	E3	OP1	OP2	OP3	NUM	<UNUM>	'+'	'-'	'*'	'/'	'^'	'('	')'	eof
S0	S1	S2	S3	S4				S5	S7		S8				S6		
S1					S9					S10	S11						accept
S2						S12				R[2]	R[2]	S13	S14		R[2]	R[2]	
S3										R[4]	R[4]	R[4]	R[4]			R[4]	R[4]
S4							S15			R[6]	R[6]	R[6]	R[6]	S16		R[6]	R[6]
S5										R[7]	R[7]	R[7]	R[7]	R[7]		R[7]	R[7]
S6	S17	S2	S3	S4				S5	S7		S8				S6		
S7										R[9]	R[9]	R[9]	R[9]	R[9]		R[9]	R[9]
S8									S18								
S9		S19	S3	S4				S5	S7		S8				S6		
S10									R[11]		R[11]				R[11]		
S11									R[12]		R[12]				R[12]		
S12			S20	S4				S5	S7		S8				S6		
S13									R[13]		R[13]				R[13]		
S14									R[14]		R[14]				R[14]		
S15			S21	S4				S5	S7		S8				S6		
S16									R[15]		R[15]				R[15]		
S17					S9					S10	S11					S22	
S18									R[10]	R[10]				R[10]		R[10]	R[10]
S19						S12				R[1]	R[1]	S13	S14			R[1]	R[1]
S20										R[3]	R[3]	R[3]	R[3]			R[3]	R[3]
S21										R[5]	R[5]					R[5]	R[5]
S22										R[8]	R[8]	R[8]	R[8]			R[8]	R[8]

## 5.2.3 应用 SLR 解析表

本节以乘法算式  $\langle \text{UNUM} \rangle \text{'*'} \langle \text{UNUM} \rangle$  的解析为例演示 SLR 解析方法。解析过程需要使用两个栈分别记录状态和符号，每次根据栈顶状态以及下一个待读入标签选择具体的操作。具体的解析过程如表 5.2 所示。

表 5.2: 应用 SLR 解析表解析乘法算式  $\langle \text{UNUM} \rangle '*' \langle \text{UNUM} \rangle$

状态栈	符号栈	待读入标签	操作
S0		$\langle \text{UNUM} \rangle '*' \langle \text{UNUM} \rangle \langle \text{eof} \rangle$	shift $\langle \text{UNUM} \rangle$ , goto S7
S0,S7	$\langle \text{UNUM} \rangle$	$'*' \langle \text{UNUM} \rangle \langle \text{eof} \rangle$	Reduce [9], back to S0, goto S5
S0,S5	NUM	$'*' \langle \text{UNUM} \rangle \langle \text{eof} \rangle$	Reduce [7], back to S0, goto S4
S0,S4	E3	$'*' \langle \text{UNUM} \rangle \langle \text{eof} \rangle$	Reduce [6], back to S0, goto S3
S0,S3	E2	$'*' \langle \text{UNUM} \rangle \langle \text{eof} \rangle$	Reduce [4], back to S0, goto S2
S0,S2	E1	$'*' \langle \text{UNUM} \rangle \langle \text{eof} \rangle$	Shift $'*' \langle \text{UNUM} \rangle$ , goto S13
S0,S2,S13	E1 $'*' \langle \text{UNUM} \rangle$	$\langle \text{UNUM} \rangle \langle \text{eof} \rangle$	Reduce [13], back to S2, goto S12
S0,S2,S12	E1 OP2	$\langle \text{UNUM} \rangle \langle \text{eof} \rangle$	Shift $\langle \text{UNUM} \rangle$ , goto S7
S0,S2,S12,S7	E1 OP2 $\langle \text{UNUM} \rangle$	$\langle \text{eof} \rangle$	Reduce [9], back to S12, goto S5
S0,S2,S12,S5	E1 OP2 NUM	$\langle \text{eof} \rangle$	Reduce [7], back to S12, goto S4
S0,S2,S12,S4	E1 OP2 E3	$\langle \text{eof} \rangle$	Reduce [6], back to S12, goto S20
S0,S2,S12,S20	E1 OP2 E2	$\langle \text{eof} \rangle$	Reduce [3], back to S0, goto S2
S0,S2	E1	$\langle \text{eof} \rangle$	Reduce [2], back to S0, goto S1
S0,S1	E	$\langle \text{eof} \rangle$	accept

## 5.3 更多 LR 解析方法

SLR 文法的能力比较有限, 如果 SLR 解析表的单元格如果存在多个操作选项, 则不适合采用 SLR 解析方法。LR(1) 是一种典型的能力更强的自底向下解析方法, 相比 SLR 其增强的方法是在构造 LR(1) 自动机时即考虑每条规范项的 Follow 信息。如果两个规范族相同, 但其中某条规范项的 Follow 信息不同, 则创建一个新的 LR(1) 自动机状态, 通过增加自动机状态有效避免操作冲突。为减小 LR(1) 状态增加带来的副作用, LALR 将规范族相同, 但 Follow 信息不同的状态合并。LALR 相比 SLR 在 Follow 信息的使用上更为精准。

虽然 LR(1) 的能力很强, 但依然无法应对所有的上下文无关文法。常见的通用上下文无关文法解析方法包括 GLR 和 CYK。GLR(Generalized LR) [1] 是 LR 解析方法的正交扩展, 即在出现冲突时广序遍历所有可能的解析方案, 可与 LALR、LR(1) 等方法搭配。CYK(Cocke-Younger-Kasami) [2] 则是有别于 LR 的一种采用动态规划思想的解析方法。

## 练习

- 已知下列上下文无关文法规则, 为其构造 SLR 解析表。

$$\begin{aligned}
 \text{Regex} &\mapsto \text{Regex } '|' \text{Concat} \\
 \text{Regex} &\mapsto \text{Concat} \\
 \text{Concat} &\mapsto \text{Concat Closure} \\
 \text{Concat} &\mapsto \text{Closure} \\
 \text{Closure} &\mapsto \text{Closure } '*' \\
 \text{Closure} &\mapsto \text{Item} \\
 \text{Item} &\mapsto '(' \text{Regex } ')' \\
 \text{Item} &\mapsto \langle \text{Char} \rangle
 \end{aligned} \tag{5.3}$$

2. (多选题) 上述文法属于:

(a) LL(1)

(b) SLR

(c) LALR

(d) LR(1)

3. LL(1) 语法一定是 LR(1) 吗? 为什么?

4. LL(1) 一定是 LALR(1) 吗? 为什么?

## Bibliography

- [1] Masaru Tomita. "An Efficient Context-Free Parsing Algorithm for Natural Languages." In IJCAI, vol. 85, pp. 756-764. 1985.
- [2] Daniel H. Younger. "Recognition and parsing of context-free languages in time  $n^3$ ." Information and control 10, no. 2 (1967): 189-208.