形式语言与自动机理论

上下文无关文法

王春宇

计算机科学与技术学院
哈尔滨工业大学

上下文无关文法

- 上下文无关文法
  - 形式定义
  - 归约和派生
  - 最左派生和最右派生
  - 文法的语言
- 语法分析树
- 文法和语言的歧义性
- 文法的化简与范式

自然语言的文法

$$
\begin{aligned}
\langle \textit{sentence} \rangle &\to \langle \textit{noun-phrase} \rangle \langle \textit{verb-phrase} \rangle \\
\langle \textit{noun-phrase} \rangle &\to \langle \textit{article} \rangle \langle \textit{noun} \rangle \mid \langle \textit{article} \rangle \langle \textit{adjective} \rangle \langle \textit{noun} \rangle \\
\langle \textit{verb-phrase} \rangle &\to \langle \textit{verb} \rangle \mid \langle \textit{verb} \rangle \langle \textit{noun-phrase} \rangle \\
\langle \textit{article} \rangle &\to \texttt{a} \mid \texttt{the} \\
\langle \textit{noun} \rangle &\to \texttt{boy} \mid \texttt{girl} \mid \texttt{cat} \\
\langle \textit{adjective} \rangle &\to \texttt{big} \mid \texttt{small} \mid \texttt{blue} \\
\langle \textit{verb} \rangle &\to \texttt{sees} \mid \texttt{likes} \\
&\cdots
\end{aligned}
$$

自然语言的文法

使用文法规则产生句子:

$$
\begin{aligned}
\langle \textit{sentence} \rangle &\Rightarrow \langle \textit{noun-phrase} \rangle \langle \textit{verb-phrase} \rangle \\
&\Rightarrow \langle \textit{article} \rangle \langle \textit{noun} \rangle \langle \textit{verb-phrase} \rangle \\
&\Rightarrow \langle \textit{article} \rangle \langle \textit{noun} \rangle \langle \textit{verb} \rangle \langle \textit{noun-phrase} \rangle \\
&\Rightarrow \langle \textit{article} \rangle \langle \textit{noun} \rangle \langle \textit{verb} \rangle \langle \textit{article} \rangle \langle \textit{adjective} \rangle \langle \textit{noun} \rangle \\
&\Rightarrow \texttt{the } \langle \textit{noun} \rangle \langle \textit{verb} \rangle \langle \textit{article} \rangle \langle \textit{adjective} \rangle \langle \textit{noun} \rangle \\
&\Rightarrow \texttt{the girl } \langle \textit{verb} \rangle \langle \textit{article} \rangle \langle \textit{adjective} \rangle \langle \textit{noun} \rangle \\
&\Rightarrow \cdots \\
&\Rightarrow \texttt{the girl sees a blue cat}
\end{aligned}
$$

定义

如果字符串 $w \in \Sigma^*$ 满足

$$w = w^R,$$

则称字符串 w 为回文 *(palindrome)*.

定义

如果语言 L 中的字符串都是回文, 则称 L 为回文语言

$$L = \{w \in \Sigma^* \mid w = w^R\}.$$

- ε, 010, 0000, *radar*, *racecar*, *drawkward*
- A man, a plan, a canal — Panama
- 僧游云隐寺, 寺隐云游僧

例 1. 字母表 $\Sigma = \{0, 1\}$ 上的回文语言

$$L_{\text{pal}} = \{w \in \{0,1\}^* \mid w = w^R\}.$$

- 很容易证明是 L_{pal} 是非正则的. 但如何表示呢?
- 可使用递归的方式来定义:
  1. 首先 ε, 0, 1 都是回文
  2. 如果 w 是回文, $0w0$ 和 $1w1$ 也是回文
- 使用嵌套定义表示这种递归结构:

$$
\begin{aligned}
&A \to \varepsilon \qquad && A \to 0A0 \\
&A \to 0 && A \to 1A1 \\
&A \to 1 &&
\end{aligned}
$$

上下文无关文法的形式定义

定义

上下文无关文法(*CFG*, *Context-Free Grammar*, 简称文法) G 是一个四元组

$$G = (V, T, P, S),$$

1. V: 变元的有穷集, 变元也称为非终结符或语法范畴;
2. T: 终结符的有穷集, 且 $V \cap T = \varnothing$;
3. P: 产生式的有穷集, 每个产生式包括:
   1. 一个变元, 称为产生式的头或左部;
   2. 一个产生式符号 $\rightarrow$, 读作定义为;
   3. 一个 $(V \cup T)^*$ 中的符号串, 称为体或右部;
4. $S \in V$: 初始符号, 文法开始的地方.

- 产生式 $A \to \alpha$, 读作 A 定义为 α
- 如果有多个 A 的产生式

$$A \to \alpha_1,\ A \to \alpha_2,\ \cdots,\ A \to \alpha_n$$

  可简写为

$$A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$$

- 文法中变元 A 的全体产生式, 称为 A 产生式

续例 1. 回文语言 $L_{\text{pal}} = \{w \in \{0,1\}^* \mid w = w^R\}$ 的文法可设计为

$$G = (\{A\}, \{0,1\}, \{A \to \varepsilon \mid 0 \mid 1 \mid 0A0 \mid 1A1\}, A).$$

字符使用的一般约定

- 终结符: $0, 1, \ldots,\ a, b, \ldots$
- 终结符串: $\ldots, w, x, y, z$
- 非终结符: $S, A, B, \ldots$
- 终结符或非终结符: $\ldots, X, Y, Z$
- 终结符或非终结符组成的串: $\alpha, \beta, \gamma, \ldots$

例 2. 简化版的算数表达式:

- 运算只有“加”和“乘” $(+, *)$, 参数仅为标识符;
- 标识符: 以 $\{a, b\}$ 开头由 $\{a, b, 0, 1\}$ 组成的字符串.

这样的表达式集合可用文法 G_{exp} 表示

$$G_{\text{exp}} = (\{E, I\}, \{a, b, 0, 1, +, *, (,)\}, P, E),$$

其中产生式集 P 中有 10 条产生式

1. $E \to I$	5. $I \to a$	9. $I \to I0$
2. $E \to E + E$	6. $I \to b$	10. $I \to I1$
3. $E \to E * E$	7. $I \to Ia$	
4. $E \to (E)$	8. $I \to Ib$	

注意, 变元 I 所定义的标识符集合, 刚好是 $(\mathbf{a} + \mathbf{b})(\mathbf{a} + \mathbf{b} + \mathbf{0} + \mathbf{1})^*$.

归约和派生

非形式定义

从字符串到文法变元的分析过程, 称为**递归推理**或**归约**;
从文法变元到字符串的分析过程, 称为**推导**或**派生**.

$$
\begin{tikzcd}
L \arrow[dd, bend right=60, "\text{派生}"'] \\
\text{文法 } G \\
w \arrow[uu, bend right=60, "\text{归约}"']
\end{tikzcd}
$$

- 归约: 自底向上, 由产生式的体向头的分析
- 派生: 自顶向下, 由产生式的头向体分析

续例 2. 用算数表达式文法 G_{exp}, 将 $a*(a+b00)$ 归约的过程.

1. $E \to I$
2. $E \to E+E$
3. $E \to E*E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

	串归约到变元		应用产生式	重用结果
(1)	a	I	5. $I \to a$	–
(2)	b	I	5. $I \to b$	–
(3)	$b0$	I	9. $I \to I0$	(2)
(4)	$b00$	I	9. $I \to I0$	(3)
(5)	a	E	1. $E \to I$	(1)
(6)	$b00$	E	1. $E \to I$	(4)
(7)	$a+b00$	E	2. $E \to E+E$	(5), (6)
(8)	$(a+b00)$	E	4. $E \to (E)$	(7)
(9)	$a*(a+b00)$	E	3. $E \to E*E$	(5), (8)

派生和归约的形式定义

定义

若 CFG $G = (V, T, P, S)$, 设 $\alpha, \beta, \gamma \in (V \cup T)^*$, $A \in V$, $A \to \gamma \in P$, 那么称在 G 中由 $\alpha A \beta$ 可派生出 $\alpha\gamma\beta$, 记为

$$\alpha A \beta \underset{G}{\Rightarrow} \alpha\gamma\beta.$$

相应的, 称 $\alpha\gamma\beta$ 可归约为 $\alpha A \beta$.

- $\alpha A \beta \underset{G}{\Rightarrow} \alpha\gamma\beta$, 即用 $A \to \gamma$ 的右部 γ 替换串 $\alpha A \beta$ 中变元 A 得到串 $\alpha\gamma\beta$
- 如果语境中 G 是已知的, 可省略, 记为 $\alpha A \beta \Rightarrow \alpha\gamma\beta$

- 设 $\alpha_1, \ldots, \alpha_m \in (V \cup T)^*$, $m \geq 1$, 对 $i = 1, \ldots, m-1$ 如果有

$$\alpha_i \underset{G}{\Rightarrow} \alpha_{i+1}$$

成立, 即 α_1 经过零步或多步派生可得到 α_m

$$\alpha_1 \underset{G}{\Rightarrow} \alpha_2 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} \alpha_{m-1} \underset{G}{\Rightarrow} \alpha_m,$$

那么, 记为

$$\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \alpha_m.$$

- 若 α 派生出 β 刚好经过了 i 步, 可记为

$$\alpha \overset{i}{\underset{G}{\Rightarrow}} \beta.$$

续例 2. 算数表达式 $a*(a+b00)$ 在文法 G_{exp} 中的派生过程.

$$
\begin{aligned}
E &\Rightarrow E*E \Rightarrow E*(E) \Rightarrow I*(E) \\
&\Rightarrow I*(E+E) \Rightarrow I*(E+I) \Rightarrow I*(I+I) \\
&\Rightarrow I*(a+I) \Rightarrow a*(a+I) \Rightarrow a*(a+I0) \\
&\Rightarrow a*(a+I00) \Rightarrow a*(a+b00)
\end{aligned}
$$

最左派生和最右派生

定义

为限制派生的随意性，要求只替换符号串中最左边变元的派生过程，称为最左派生，记为

$$\underset{\mathrm{lm}}{\Rightarrow}, \quad \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}},$$

只替换最右的，称为最右派生，记为

$$\underset{\mathrm{rm}}{\Rightarrow}, \quad \overset{*}{\underset{\mathrm{rm}}{\Rightarrow}}.$$

- 任何派生都有等价的最左派生和最右派生

$$A \overset{*}{\Rightarrow} w \text{ 当且仅当 } A \overset{*}{\underset{\mathrm{lm}}{\Rightarrow}} w \text{ 当且仅当 } A \overset{*}{\underset{\mathrm{rm}}{\Rightarrow}} w.$$

续例 2. 表达式 $a * (a + a)$ 在 G_{exp} 中的最左派生和最右派生分别为:

1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

$$
\begin{aligned}
E &\underset{\text{lm}}{\Rightarrow} E * E \\
&\underset{\text{lm}}{\Rightarrow} I * E \\
&\underset{\text{lm}}{\Rightarrow} a * E \\
&\underset{\text{lm}}{\Rightarrow} a * (E) \\
&\underset{\text{lm}}{\Rightarrow} a * (E + E) \\
&\underset{\text{lm}}{\Rightarrow} a * (I + E) \\
&\underset{\text{lm}}{\Rightarrow} a * (a + E) \\
&\underset{\text{lm}}{\Rightarrow} a * (a + I) \\
&\underset{\text{lm}}{\Rightarrow} a * (a + a)
\end{aligned}
$$

$$
\begin{aligned}
E &\underset{\text{rm}}{\Rightarrow} E * E \\
&\underset{\text{rm}}{\Rightarrow} E * (E) \\
&\underset{\text{rm}}{\Rightarrow} E * (E + E) \\
&\underset{\text{rm}}{\Rightarrow} E * (E + I) \\
&\underset{\text{rm}}{\Rightarrow} E * (E + a) \\
&\underset{\text{rm}}{\Rightarrow} E * (I + a) \\
&\underset{\text{rm}}{\Rightarrow} E * (a + a) \\
&\underset{\text{rm}}{\Rightarrow} I * (a + a) \\
&\underset{\text{rm}}{\Rightarrow} a * (a + a)
\end{aligned}
$$

文法的语言

定义

$CFG\ G = (V, T, P, S)$ 的**语言**定义为

$$\mathbf{L}(G) = \{w \mid w \in T^*,\ S \overset{*}{\underset{G}{\Rightarrow}} w\}.$$

那么符号串 w 在 $\mathbf{L}(G)$ 中, 要满足:

1. w 仅由终结符组成;
2. 初始符号 S 能派生出 w.

上下文无关语言

定义

语言 L 是某个 CFG G 定义的语言，即 $L = \mathbf{L}(G)$，则称 L 为上下文无关语言 *(CFL, Context-Free Language)*.

- 上下文无关是指在文法派生的每一步

$$\alpha A\beta \Rightarrow \alpha\gamma\beta,$$

符号串 γ 仅根据 A 的产生式派生，而无需依赖 A 的上下文 α 和 β.

文法的等价性

定义

如果有两个文法 $CFG\ G_1$ 和 $CFG\ G_2$, 满足

$$\mathbf{L}(G_1) = \mathbf{L}(G_2),$$

则称 G_1 和 G_2 是等价的.

句型

定义

若 CFG $G=(V,T,P,S)$，初始符号 S 派生出来的符号串，称为 G 的句型，即

$$\alpha \in (V \cup T)^* \text{ 且 } S \overset{*}{\Rightarrow} \alpha.$$

如果 $S \underset{\text{lm}}{\overset{*}{\Rightarrow}} \alpha$，称 α 为左句型.
如果 $S \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha$，称 α 为右句型.

- 只含有终结符的句型，也称为 G 的句子
- 而 $\mathbf{L}(G)$ 就是文法 G 全部的句子

例 3. 给出语言 $L = \{w \in \{0,1\}^* \mid w \text{ contains at least three 1s}\}$ 的文法.

解: $S \to A1A1A1A$, $A \to 0A \mid 1A \mid \varepsilon$

例 4. 描述 CFG $G = (\{S\}, \{a, b\}, \{S \to aSb, S \to ab\}, S)$ 定义的语言?

解: $\mathbf{L}(G) = \{a^n b^n \mid n \geq 1\}$, 因为 $S \Rightarrow aSb \Rightarrow \cdots \Rightarrow a^{n-1}Sb^{n-1} \Rightarrow a^n b^n$.

例 5. 请为语言 $L = \{0^n 1^m \mid n \neq m\}$ 设计文法.

解:

$$
\begin{aligned}
&S \to AC \mid CB \qquad &&A \to A0 \mid 0 \\
&C \to 0C1 \mid \varepsilon \qquad &&B \to 1B \mid 1
\end{aligned}
$$

例 6. 设计 $L_{\mathrm{eq}} = \{w \in \{0,1\}^* \mid w$ 中 0 和 1 个数相等$\}$ 的文法.

解 1: $S \to 0S1 \mid 1S0 \mid SS \mid \varepsilon$, 寻找递归结构, 用变量构造递归结构;
解 2: $S \to S0S1S \mid S1S0S \mid \varepsilon$, “目标串”这样构成, 由变量定义变量.

程序设计语言的文法定义

- C — ISO C 1999 definition

  ...

  selection-statement:

  `if` `(` *expression* `)` *statement*

  `if` `(` *expression* `)` *statement* `else` *statement*

  `switch` `(` *expression* `)` *statement*

  ...

- Python — Full Grammar specification

  ...

```
try_stmt: ('try' ':' suite
           ((except_clause ':' suite)+
            ['else' ':' suite]
            ['finally' ':' suite] |
           'finally' ':' suite))
```

  ...