形式语言与自动机理论

上下文无关文法

王春宇
chunyu@hit.edu.cn

计算学部
哈尔滨工业大学

2021 年 4 月

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*).

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

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*).

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

定义

如果语言 L 中的字符串都是回文, 则称 L 为**回文语言**

$$L = \left\{ w \in \Sigma^* \,\middle|\, w = w^R \right\}.$$

例 1. 字母表 $\Sigma = \{0, 1\}$ 上的回文语言

$$L_{\text{pal}} = \left\{ w \in \{0,1\}^* \;\middle|\; w = w^R \right\}.$$

- 很容易证明是 L_{pal} 是非正则的. 但如何表示呢?

例 1. 字母表 $\Sigma = \{0,1\}$ 上的回文语言

$$L_{\text{pal}} = \left\{ w \in \{0,1\}^* \;\middle|\; w = w^R \right\}.$$

- 很容易证明是 L_{pal} 是非正则的. 但如何表示呢?
- 可以用“递归”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.

例 1. 字母表 $\Sigma = \{0, 1\}$ 上的回文语言

$$L_{\mathrm{pal}} = \left\{ w \in \{0,1\}^* \;\middle|\; w = w^R \right\}.$$

- 很容易证明是 L_{pal} 是非正则的. 但如何表示呢?
- 可以用 “递归” 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:

$$A \to \varepsilon \qquad A \to 0A0$$
$$A \to 0 \qquad A \to 1A1$$
$$A \to 1$$

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 $L_{\text{pal}} = \left\{ w \in \{0,1\}^* \,\middle|\, w = w^R \right\}$ 的文法可设计为

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

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.

例 2. 简化版算数表达式:

- 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

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

其中产生式集 P 中有 10 条产生式

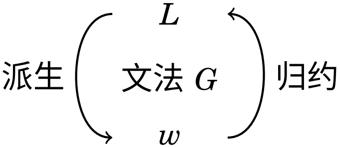
1. $E \to I$	5. $I \to a$	9. $I \to I0$
2. $E \to E + E$	6. $I \to b$	10. $I \to I1$
3. $E \to E * E$	7. $I \to Ia$	
4. $E \to (E)$	8. $I \to Ib$	

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

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**.



- 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. $I \to I1$

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

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

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 $L = \left\{ w \in \{0,1\}^* \,\middle|\, w \text{ contains at least three 1s} \right\}$ 的文法.

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

例 4. 描述 CFG $G = \big(\{S\}, \{a, b\}, \{S \to aSb, S \to ab\}, S\big)$ 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:

$$S \to AC \mid CB \qquad A \to A0 \mid 0$$
$$C \to 0C1 \mid \varepsilon \qquad B \to 1B \mid 1$$

例 6. 设计 $L_{\mathrm{eq}} = \left\{ w \in \{0,1\}^* \;\middle|\; w \text{ 中 } 0 \text{ 和 } 1 \text{ 个数相等} \right\}$ 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.

例 7. 设计 $L_{j\geq 2i}=\{\,a^i b^j \mid j\geq 2i\,\}$ 的文法.

解:

$$\begin{aligned} S &\to AB \\ A &\to aAbb \mid \varepsilon \\ B &\to bB \mid \varepsilon \end{aligned}$$

或

$$\begin{aligned} S &\to aSbb \mid B \\ B &\to \varepsilon \mid bB \end{aligned}$$

课堂练习. Design CFG for strings in $\{0,1\}^*$ in which the number of 0s is greater than or equal to the number of 1s.

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

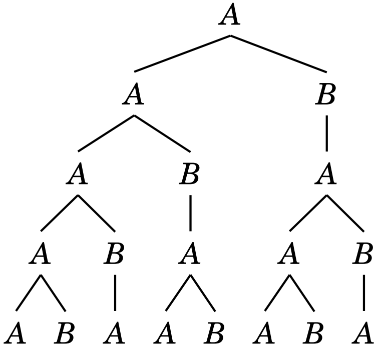
  ...

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

  ...

细胞生长的数学模型

- Lindenmayer System [1] – 自然递归规则导致自相似性
  - 如海藻生长 (或兔子繁殖): $A \to AB$, $B \to A$



- Fibonacci 数列 $1\ 2\ 3\ 5\ 8\ 13\ 21\ \cdots$

$$n = 0 : A$$
$$n = 1 : AB$$
$$n = 2 : ABA$$
$$n = 3 : ABAAB$$
$$n = 4 : ABAABABA$$
$$n = 5 : ABAABABAABAAB$$
$$\cdots$$

[1]由荷兰生物和植物学家 Aristid Lindenmayer 于 1968 年提出

例 8. [Exe. 5.1.3] Show that every regular laugnage is a context-free laugnage.

例 8. [Exe. 5.1.3] Show that every regular laugnage is a context-free laugnage.

证明：对正则表达式 R 中运算符的个数 n 进行归纳.
归纳基础: 当 $n=0$ 时, R 只能是 $\boldsymbol{\varepsilon}$, $\varnothing$ 或 $\mathbf{a}$ $(a \in \Sigma)$, 可以构造仅有一条产生式的文法 $S \to \varepsilon$, $S \to \varnothing$ 或 $S \to a$ 得到.
归纳递推: 假设当 $n \le m$ 时成立. 当 $n = m+1$ 时, R 的形式只能由表达式 R_1 和 R_2 由连接、并或闭包形成：

- 若 $R = R_1 + R_2$, 则 R_1 和 R_2 中运算符都不超过 m, 所以都存在文法 G_1 和 G_2, 分别开始于 S_1 和 S_2, 只需构造新产生式和开始符号 $S \to S_1|S_2$, 连同 G_1 和 G_2 的产生式, 构成 R 的文法;
- 若 $R = R_1R_2$, 则同理构造 $S \to S_1S_2$ 即可;
- 若 $R = R_1^*$, 则构造 $S \to SS_1 \mid \varepsilon$ 即可.

且每种构造, 文法的语言与该表达式的语言等价. □

例 9. [Exe. 5.1.5] Let $T = \{0, 1, (,), +, *, \varnothing, e\}$. We may think of T as the set of symbols used by regular expressions over the alphabet $\{0, 1\}$; the only difference is that we use e for symbol ε, to avoid potential confusion in what follows. Your task is to design a CFG with set of terminals T that generates exactly the regular expressions with alphabet $\{0, 1\}$.

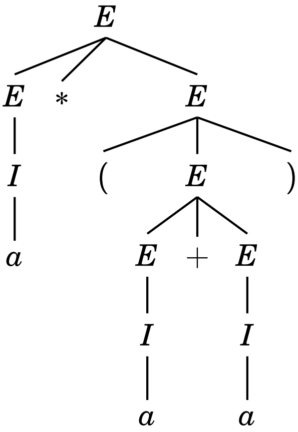
解: $S \to S + S \mid SS \mid S^* \mid (S) \mid 0 \mid 1 \mid \varnothing \mid e$.

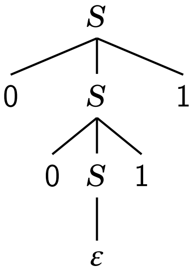
上下文无关文法

- 上下文无关文法
- 语法分析树
  - 形式定义
  - 语法树和派生的等价性
- 文法和语言的歧义性
- 文法的化简与范式

派生或归约的过程可以表示成树形结构.

- 例 2 文法 G_{exp} 中推导算数表达式 $a * (a + a)$ 的过程
- 例 6 中语言 L_{eq} 的文法中推导 0011 的过程





语法分析树的形式定义

定义

CFG $G = (V, T, P, S)$ 的**语法分析树**(语法树或派生树) 为:

1. 每个内节点标记为 V 中的变元符号;
2. 每个叶节点标记为 $V \cup T \cup \{\varepsilon\}$ 中的符号;
3. 如果某内节点标记是 A, 其子节点从左至右分别为

$$X_1,\ X_2,\ \cdots,\ X_n$$

那么

$$A \to X_1 X_2 \cdots X_n \in P,$$

若有 $X_i = \varepsilon$, 则 ε 是 A 唯一子节点, 且 $A \to \varepsilon \in P$.

定义

语法树的全部叶节点从左到右连接起来, 称为该树的**产物**或结果.

- 如果树根节点是初始符号 S, 叶节点是终结符或 ε, 那么该树的产物属于 $\mathbf{L}(G)$.

定义

语法树中标记为 A 的内节点及其全部子孙节点构成的子树, 称为 **A 子树**.

语法分析树和派生的等价性

定理 17

CFG $G = (V, T, P, S)$ 且 $A \in V$, 那么文法 G 中

$$A \overset{*}{\Rightarrow} \alpha$$

当且仅当 G 中存在以 A 为根节点产物为 α 的语法树.

证明: [充分性] 对 $A \overset{j}{\Rightarrow} \alpha$ 的步骤数 j 归纳证明.

证明: [充分性] 对 $A \stackrel{j}{\Rightarrow} \alpha$ 的步骤数 j 归纳证明.

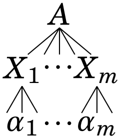
归纳基础: 当 $j = 1$ 时, 即 $A \Rightarrow \alpha$, 那么有 $A \to \alpha \in P$, 可构造 $\begin{array}{c} A \\ \diagup | \diagdown \\ \alpha \end{array}$.

归纳递推: 假设 $j \le n$ 时命题成立. 当 $j = n+1$ 时, $A \xRightarrow{n+1} \alpha$ 的派生过程为

$$A \Rightarrow X_1 \cdots X_m \stackrel{n}{\Rightarrow} \alpha_1 \cdots \alpha_m = \alpha,$$

即第 1 步一定由某产生式 $A \to X_1 X_2 \cdots X_m \in P$ 派生.

而 X_i 若非终结符, 一定有 $X_i \stackrel{*}{\Rightarrow} \alpha_i$ 且不超过 n 步, 由归纳假设存在语法树 $\begin{array}{c} X_i \\ \diagup | \diagdown \\ \alpha_i \end{array}$. 因此可以构造以 A 为根, 以 X_i 为子树 (或叶子) 的语法树, 其产物刚好为 α.



[必要性] 对语法分析树的内节点数 j 归纳证明.

[必要性] 对语法分析树的内节点数 j 归纳证明.

归纳基础: 当 $j=1$ 时, 即 $\begin{array}{c}A\\ \bigwedge\\ \alpha\end{array}$, A 必为根, 则 $A \to \alpha \in P$, 所以 $A \overset{*}{\Rightarrow} \alpha$.

归纳递推: 假设 $j \le n$ 时命题成立. 当 $j=n+1$ 时, 根节点 A 的儿子依次为 $X_1, X_2, \ldots, X_m$, 则

$$A \to X_1 \cdots X_m \in P, \text{ 且 } A \Rightarrow X_1 \cdots X_m.$$

而 X_i 子树 (或叶子) 内节点数都不超过 n, 由归纳假设有

$$X_i \overset{*}{\Rightarrow} \alpha_i,$$

从左至右连接 α_i, 刚好为树的产物 α, 所以有

$$X_1 X_2 \cdots X_m \overset{*}{\Rightarrow} \alpha_1 X_2 \cdots X_m \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} \alpha_1 \alpha_2 \cdots \alpha_m = \alpha.$$

因此 $A \overset{*}{\Rightarrow} \alpha$ 命题成立. $\square$

语法树唯一确定最左 (右) 派生

- 每棵语法分析树都有唯一的最左 (右) 派生
- 给定 CFG $G = (V, T, P, S)$, $A \in V$, 以下命题等价:
  1. 通过递归推理, 确定串 w 在变元 A 的语言中.
  2. 存在以 A 为根节点, 产物为 w 的语法分析树.
  3. $A \overset{*}{\Rightarrow} w$.
  4. $A \underset{\text{lm}}{\overset{*}{\Rightarrow}} w$.
  5. $A \underset{\text{rm}}{\overset{*}{\Rightarrow}} w$.

例 10. [Exe. 5.2.2] Suppose that G is a CFG without any productions that have ε as the right side. If w is in $L(G)$, the length of w is n, and w has a derivation of m steps, show that w has a parse tree with $n+m$ nodes.

例 11. [Exe. 5.2.3] Suppose all is as in Exercise 5.2.2, but G may have some productions with ε as the right side. Show that a parse tree for a string w other than ε may have as many as $n+2m-1$ nodes, but no more.

上下文无关文法

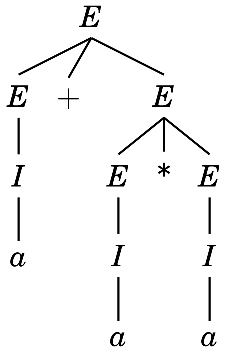
- 上下文无关文法
- 语法分析树
- 文法和语言的歧义性
  - 文法歧义性的消除
  - 语言的固有歧义性
- 文法的化简与范式

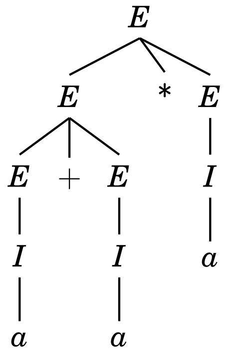
文法的歧义性

定义

如果 CFG G 使某些符号串有两棵不同的语法分析树, 称文法 G 是**歧义的**.

续例2. 算数表达式的文法 G_{exp} 中, 对句型 $a+a*a$ 有下面两棵语法分析树:





(1) $E \Rightarrow E+E$

$\Rightarrow E+E*E$

$\overset{*}{\Rightarrow} a+a*a$

(2) $E \Rightarrow E*E$

$\Rightarrow E+E*E$

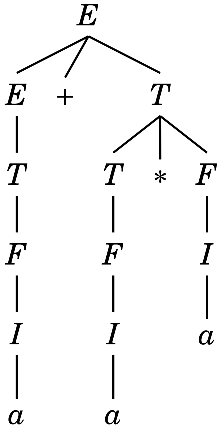
$\overset{*}{\Rightarrow} a+a*a$

文法歧义性的消除

有些文法的歧义性, 可以通过重新设计文法来消除.

续例 2. 文法 G_{exp} 重新设计为文法 $G_{\text{exp}'}$ 可消除歧义.

$$
\begin{aligned}
E &\to E + T \mid T \\
T &\to T * F \mid F \\
F &\to (E) \mid I \\
I &\to a \\
I &\to b \\
I &\to Ia \\
I &\to Ib \\
I &\to I0 \\
I &\to I1
\end{aligned}
$$



语言的固有歧义性

定义

定义同样的语言可以有多个文法，如果 *CFL* L 的所有文法都是歧义的，那么称语言 L 是固有歧义的.

- 固有歧义的语言确实存在，如语言

$$L = \{ a^i b^j c^k \mid i = j \text{ or } j = k \}$$

  中任何形为 $a^n b^n c^n$ 的串，总会有两棵语法树.
- "判定任何给定 CFG G 是否歧义" 是一个不可判定问题.

上下文无关文法

- 上下文无关文法
- 语法分析树
- 文法和语言的歧义性
- 文法的化简与范式
  - 消除无用符号
  - 消除 ε-产生式
  - 消除单元产生式
  - 乔姆斯基范式
  - 格雷巴赫范式

为什么要化简

- 典型问题: 给定 CFG G 和串 w, 判断 $w \in \mathbf{L}(G)$?
- 编译器设计和自然语言处理的基本问题之一
- 但文法的形式非常自由, 过于复杂不易于自动处理
- 以不改变语言为前提, 化简文法和限制文法的格式

例 12. 如下文法中, 有无意义的变元和产生式

$$
\begin{aligned}
S &\to 0DS1D \mid B \mid \varepsilon \\
B &\to BC1 \mid 0CBC \\
A &\to A0 \mid A1 \mid \varepsilon \\
C &\to D \\
D &\to \varepsilon
\end{aligned}
$$

文法的化简

1. 消除无用符号: 对文法定义语言没有贡献的符号
2. 消除 ε 产生式: $A \to \varepsilon$ (得到语言 $L - \{\varepsilon\}$)
3. 消除单元产生式: $A \to B$

无用符号

定义

CFG $G=(V,T,P,S)$, 符号 $X\in(V\cup T)$:

1. 如果 $S\overset{*}{\Rightarrow}\alpha X\beta$, 称 X 是可达的;
2. 如果 $\alpha X\beta\overset{*}{\Rightarrow}w\ (w\in T^*)$, 称 X 是产生的;
3. 如果 X 同时是产生的和可达的, 即

$$S\overset{*}{\Rightarrow}\alpha X\beta\overset{*}{\Rightarrow}w\ (w\in T^*),$$

则称 X 是有用的, 否则称 X 为无用符号.

消除无用符号

消除无用符号: 删除全部含有 “非产生的” 和 “非可达的” 符号的产生式

计算“产生的”符号集

1. 每个 T 中的符号都是产生的;
2. $A \to \alpha \in P$ 且 α 中符号都是产生的, 则 A 是产生的.

计算“可达的”符号集

1. 符号 S 是可达的;
2. $A \to \alpha \in P$ 且 A 是可达的, 则 α 中符号都是可达的.

定理 18

每个非空的 *CFL* 都能被一个不带无用符号的 *CFG* 定义.

注意

- 先寻找并消除全部非“产生的”符号
- 再寻找并消除全部非“可达的”符号
- 否则可能消除不完整

例 13. 消除如下文法无用符号

$$S \to AB \mid a$$
$$A \to b$$

解：先消除非产生的

$$S \to a$$
$$A \to b$$

再消除非可达的

$$S \to a$$

消除 ε-产生式

定义

文法中形如 $A \to \varepsilon$ 的产生式称为ε-产生式.
如果变元 $A \overset{*}{\Rightarrow} \varepsilon$, 称 A 是可空的.

- ε-产生式在文法定义语言时, 除产生空串外没有其他帮助
- 对于 CFL L, 消除其文法中全部的 ε-产生式后, 得到语言 $L - \{\varepsilon\}$

确定“可空变元”

1. 如果 $A \to \varepsilon$, 则 A 是可空的;
2. 如果 $B \to \alpha$ 且 α 中的每个符号都是可空的, 则 B 是可空的.

替换产生式

将含有可空变元的一条产生式 $A \to X_1X_2\cdots X_n$,
用一组产生式 $A \to Y_1Y_2\cdots Y_n$ 代替, 其中:

1. 若 X_i 不是可空的, Y_i 为 X_i;
2. 若 X_i 是可空的, Y_i 为 X_i 或 ε;
3. 但 Y_i 不能全为 ε.

定理 19

任何 CFG G, 都存在一个不带无用符号和 ε-产生式的 CFG G', 使 $\mathbf{L}(G') = \mathbf{L}(G) - \{\varepsilon\}$.

例 14. 消除 CFG $G = (\{S, A, B\}, \{a, b\}, P, S)$ 的 ε-产生式.

$$S \to AB$$
$$A \to AaA \mid \varepsilon$$
$$B \to BbB \mid \varepsilon$$

解: CFG G' 为

$$S \to AB \mid A \mid B$$
$$A \to AaA \mid Aa \mid aA \mid a$$
$$B \to BbB \mid Bb \mid bB \mid b$$

消除单元产生式

确定“单元对”

如果有 $A \overset{*}{\Rightarrow} B$, 则称 $[A,B]$ 为单元对.

1. $A \to B \in P$, 则 $[A,B]$ 是单元对;
2. 若 $[A,B]$ 和 $[B,C]$ 都是单元对, 则 $[A,C]$ 是单元对.

消除单元产生式

1. 删除全部形为 $A \to B$ 的单元产生式;
2. 对每个单元对 $[A,B]$, 将 B 的产生式复制给 A.

定理 20

每个不带 ε 的 CFL 都可由一个不带无用符号, ε-产生式和单元产生式的文法定义.

例 15. 消除文法的单元产生式

$$
\begin{aligned}
S &\to A \mid B \mid 0S1 \\
A &\to 0A \mid 0 \\
B &\to 1B \mid 1
\end{aligned}
$$

解: 单位对为 $[S,A]$ 和 $[S,B]$, 带入得:

$$
\begin{aligned}
S &\to 0S1 & \qquad A &\to 0A \mid 0 \\
S &\to 0A \mid 0 & \qquad B &\to 1B \mid 1 \\
S &\to 1B \mid 1
\end{aligned}
$$

文法化简的可靠顺序

1. 消除 ε-产生式;
2. 消除单元产生式;
3. 消除非产生的无用符号;
4. 消除非可达的无用符号.

例 16. [Exe. 7.1.2] Begin with the grammar:

$$S \to ASB \mid \varepsilon$$
$$A \to aAS \mid a$$
$$B \to SbS \mid A \mid bb$$

1. Eliminate $\varepsilon-$productions.
2. Eliminate any unit productions in the resulting grammar.
3. Eliminate any useless symbols in the resulting grammar.

限制文法格式

将任意形式的文法转换为:

1. 乔姆斯基范式 (CNF, Chomsky Normal Form)
2. 格雷巴赫范式 (GNF, Greibach Normal Form)

乔姆斯基范式

定理 21 (乔姆斯基范式 CNF)

每个不带 ε 的 *CFL* 都可由这样的 *CFG* G 定义, G 中每个产生式都形为

$$A \to BC \quad \text{或} \quad A \to a$$

其中 A, B 和 C 都是变元, a 是终结符.

- 利用 CNF 派生长度为 n 的串, 刚好需要 $2n-1$ 步
- 因此存在算法判断任意字符串 w 是否在给定的 CFL 中
- 利用 CNF 的 CYK 算法 — $O(n^3)$ 时间复杂度的解析算法

CFG 转为 CNF 的方法

1. 将产生式
$$A \to X_1X_2\cdots X_m \quad (m \geq 2)$$
中每个终结符 a 替换为新变元 C_a, 并增加新产生式
$$C_a \to a$$

2. 引入新变元 $D_1, D_2, \cdots, D_{m-2}$, 将产生式
$$A \to B_1B_2\cdots B_m \quad (m > 2)$$
替换为一组级联的产生式
$$\begin{gathered} A \to B_1D_1 \\ D_1 \to B_2D_2 \\ \cdots \\ D_{m-2} \to B_{m-1}B_m \end{gathered}$$

例 17. CFG $G = (\{S, A, B\}, \{a, b\}, P, S)$, 产生式集合 P 为:

$$
\begin{aligned}
S &\to bA \mid aB \\
A &\to bAA \mid aS \mid a \\
B &\to aBB \mid bS \mid b
\end{aligned}
$$

请设计等价的 CNF 文法.

解: CNF 为

$$
\begin{aligned}
&S \to C_bA \mid C_aB \\
&A \to C_aS \mid C_bD_1 \mid a \qquad D_1 \to AA \qquad C_a \to a \\
&B \to C_bS \mid C_aD_2 \mid b \qquad D_2 \to BB \qquad C_b \to b
\end{aligned}
$$

证明: 设 CFL L 不含 ε, 由定理 20, 存在不含 ε-产生式和单元产生式的等价文法 $G_1=(V,T,P,S)$. 考虑 P 中一条产生式 $A\to X_1X_2\ldots X_m$ $(m\geq 2)$.

1. 若某个 X_i 是终结符 a, 则引入新变元 C_a 和新产生式 $C_a\to a$, 并用 C_a 替换 X_i, 得文法 $G_2=(V',T,P',S)$.
2. 显然 $\mathbf{L}(G_1)\subseteq\mathbf{L}(G_2)$, 因为如果 $\alpha\underset{G_1}{\Rightarrow}\beta$, 那么 $\alpha\overset{*}{\underset{G_2}{\Rightarrow}}\beta$.
3. 用归纳法证明 $A\overset{i}{\underset{G_2}{\Rightarrow}}w\Longrightarrow A\overset{*}{\underset{G_1}{\Rightarrow}}w$, 这里的 $A\in V$, $w\in T^*$.
   1. 当 $i=1$ 时是显然的, 或者用了 P 中未修改的产生式, 或者用了被修改的产生式, 而二者都有 $A\overset{*}{\underset{G_1}{\Rightarrow}}w$.
   2. 假设当 $i\leq n$ 时命题成立. 当 $i=n+1$ 时, $A\overset{i}{\underset{G_2}{\Rightarrow}}w$ 的第 1 步, 必然使用了某个产生式 $A\to B_1B_2\cdots B_m$, 即 $A\underset{G_2}{\Rightarrow}B_1B_2\cdots B_m\overset{n}{\underset{G_2}{\Rightarrow}}w=w_1w_2\cdots w_m$ 那么, 如果 $B_i\in V'-V$, B_i 一定是对应某个终结符 a_i 的 C_{a_i}, w_i 必然是 a_i, 令 $X_i=a_i$; 如果 $B_i\in V$, $B_i\overset{*}{\underset{G_2}{\Rightarrow}}w_i$ 一定不超过 n 步, 由归纳假设, $B_i\overset{*}{\underset{G_1}{\Rightarrow}}w_i$, 那么令 $X_i=B_i$. 由 P' 的结构, $A\to X_1X_2\cdots X_m$ 是 P 的一条产生式, 所以 $A\underset{G_1}{\Rightarrow}X_1X_2\cdots X_m\overset{*}{\underset{G_1}{\Rightarrow}}w_1w_2\cdots w_m=w$.

所以 $\mathbf{L}(G_2)\subseteq\mathbf{L}(G_1)$. □

格雷巴赫范式

定理 22 (格雷巴赫范式 GNF)

每个不带 ε 的 *CFL* 都可由这样的 *CFG* G 定义, G 中每个产生式都形为

$$A \to a\alpha$$

其中 A 是变元, a 是终结符, α 是零或多个变元的串.

- GNF 每个产生式都会引入一个终结符
- 长度为 n 的串的派生恰好是 n 步

例 18. 将以下文法转换为 GNF.

$$
\begin{aligned}
&S \to AB \\
&A \to aA \mid bB \mid b \\
&B \to b
\end{aligned}
$$

解: GNF 为

$$
\begin{aligned}
&S \to aAB \mid bBB \mid bB \\
&A \to aA \mid bB \mid b \\
&B \to b
\end{aligned}
$$

直接左递归

定义

文法中形式为 $A \rightarrow A\alpha$ 的产生式, 称为直接左递归.

消除直接左递归

1. 若 A 产生式

$$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \cdots \mid \beta_m$$

其中 $\alpha_i \neq \varepsilon$, β_j 不以 A 开始;

2. 引入新变元 B, 并用如下产生式替换

$$\begin{aligned} A &\to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_m \mid \beta_1 B \mid \beta_2 B \mid \cdots \mid \beta_m B \\ B &\to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n \mid \alpha_1 B \mid \alpha_2 B \mid \cdots \mid \alpha_n B \end{aligned}$$

间接左递归

定义

文法中如果有形式为

$$
\begin{aligned}
A &\to B\alpha \mid \ldots \\
B &\to A\beta \mid \ldots
\end{aligned}
$$

的产生式, 称为间接左递归.

- 会有 $A \Rightarrow B\alpha \Rightarrow A\beta\alpha$, 无法通过代换消除递归

消除间接左递归

1. 将文法中变元重命名为 $A_1, A_2, \cdots, A_n$;
2. 通过代入, 使产生式都形如

$$A_i \to A_j\alpha$$
$$A_i \to a\alpha$$

   但要求 $i \le j$;
3. 消除直接左递归 $A_i \to A_i\beta$, 再代入其他产生式.

例 19. Convert the following grammar to GNF.

$$S \to AB$$

$$A \to BS \mid b$$

$$B \to SA \mid a$$

解:

1. 重命名变元, 代换 $i > j$ 的 A_j

$$A_1 \to A_2A_3$$

$$A_2 \to A_3A_1 \mid b$$

$$A_3 \to a \mid \cancel{A_1A_2 \mid} \cancel{A_2A_3A_2 \mid}$$

$$A_3A_1A_3A_2 \mid bA_3A_2$$

2. 消除直接左递归

$$A_1 \to A_2A_3$$

$$A_2 \to A_3A_1 \mid b$$

$$A_3 \to bA_3A_2 \mid a \mid bA_3A_2B_1 \mid aB_1$$

$$B_1 \to A_1A_3A_2 \mid A_1A_3A_2B_1$$

3. A_3 产生式代入到 A_2, A_2 产生式代入到 A_1, A_1 产生式代入 B_1

$$A_3 \to bA_3A_2 \mid a \mid bA_3A_2B_1 \mid aB_1$$

$$A_2 \to bA_3A_2A_1 \mid aA_1 \mid bA_3A_2B_1A_1 \mid aB_1A_1 \mid b$$

$$A_1 \to bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2B_1A_1A_3 \mid aB_1A_1A_3 \mid bA_3$$

$$B_1 \to bA_3A_2A_1A_3A_3A_2 \mid aA_1A_3A_3A_2 \mid bA_3A_2B_1A_1A_3A_3A_2 \mid$$

$$aB_1A_1A_3A_3A_2 \mid bA_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2B_1 \mid aA_1A_3A_3A_2B_1 \mid$$

$$bA_3A_2B_1A_1A_3A_3A_2B_1 \mid aB_1A_1A_3A_3A_2B_1 \mid bA_3A_3A_2B_1$$

GNF 引理 1

如果有文法 $G = (V, T, P, S)$, 设 $A \to \alpha_1 B \alpha_2$ 是 P 中的一个产生式, 且 $B \to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$ 是 P 中的全部 B 产生式. 将产生式 $A \to \alpha_1 B \alpha_2$ 从 P 中删除, 并增加

$$A \to \alpha_1\beta_1\alpha_2 \mid \alpha_1\beta_2\alpha_2 \mid \cdots \mid \alpha_1\beta_n\alpha_2$$

一组产生式, 得到文法 $G_1 = (V, T, P', S)$, 那么 $\mathbf{L}(G) = \mathbf{L}(G_1)$.

GNF 引理 1

如果有文法 $G=(V,T,P,S)$, 设 $A\to\alpha_1B\alpha_2$ 是 P 中的一个产生式, 且 $B\to\beta_1\mid\beta_2\mid\cdots\mid\beta_n$ 是 P 中的全部 B 产生式. 将产生式 $A\to\alpha_1B\alpha_2$ 从 P 中删除, 并增加

$$A\to\alpha_1\beta_1\alpha_2\mid\alpha_1\beta_2\alpha_2\mid\cdots\mid\alpha_1\beta_n\alpha_2$$

一组产生式, 得到文法 $G_1=(V,T,P',S)$, 那么 $\mathbf{L}(G)=\mathbf{L}(G_1)$.

证明:

1. 显然 $\mathbf{L}(G_1)\subseteq\mathbf{L}(G)$, 因为 G_1 的派生中, 如果用到了 $A\to\alpha_1\beta_i\alpha_2$, 在 G 中可以使用 $A\underset{G}{\Rightarrow}\alpha_1B\alpha_2\underset{G}{\Rightarrow}\alpha_1\beta_i\alpha_2$.
2. 而因为 $A\to\alpha_1B\alpha_2$ 是唯一在 G 中而不再 G_1 中的产生式, 每当 G 的派生中用到了 $\alpha_1B\alpha_2$ 时, 一定会在后面某一步中用到形如 $B\to\beta_i$ 的产生式来派生 B, 这两步在 G_1 中可以使用一步 $A\underset{G_1}{\Rightarrow}\alpha_1\beta_i\alpha_2$ 来代替, 所以 $\mathbf{L}(G)\subseteq\mathbf{L}(G_1)$. □

GNF 引理 2

如果有文法 $G=(V,T,P,S)$, 设带有直接左递归的 A 产生式为

$$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \cdots \mid \beta_m$$

其中 β_i 不以 A 开头. 在 V 中引入新的变元 B 并用以下产生式

$$\begin{aligned} A &\to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_m \mid \beta_1 B \mid \beta_2 B \mid \cdots \mid \beta_m B \\ B &\to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n \mid \alpha_1 B \mid \alpha_2 B \mid \cdots \mid \alpha_n B \end{aligned}$$

替换全部 A 产生式, 得到文法 $G_1=(V\cup\{B\},T,P',S)$, 那么 $\mathbf{L}(G)=\mathbf{L}(G_1)$.

证明: 在文法 G 中一系列使用 $A \to A\alpha_i$ 的最左派生, 最后必以 $A \to \beta_j$ 结束, 而这样的最左派生

$$
\begin{aligned}
A &\underset{\text{lm}}{\Rightarrow} A\alpha_{i_1} \underset{\text{lm}}{\Rightarrow} A\alpha_{i_2}\alpha_{i_1} \underset{\text{lm}}{\Rightarrow} \cdots \\
&\underset{\text{lm}}{\Rightarrow} A\alpha_{i_p}\alpha_{i_{p-1}}\ldots\alpha_{i_1} \\
&\underset{\text{lm}}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}}\ldots\alpha_{i_1},
\end{aligned}
$$

在 G_1 中可以使用一系列最右派生来代替

$$
\begin{aligned}
A &\underset{\text{rm}}{\Rightarrow} \beta_j B \underset{\text{rm}}{\Rightarrow} \beta_j\alpha_{i_p}B \underset{\text{rm}}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}}B \underset{\text{rm}}{\Rightarrow} \cdots \\
&\underset{\text{rm}}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}}\ldots\alpha_{i_2}B \\
&\underset{\text{rm}}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}}\ldots\alpha_{i_2}\alpha_{i_1}.
\end{aligned}
$$

而且, 相反的转换也成立, 因此 $\mathbf{L}(G) = \mathbf{L}(G_1)$. □

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

chunyu@hit.edu.cn
http://nclab.net/~chunyu