上下文无关语言上的递归函数
——Ⅱ. CFPRF及CFRF定义的合宜性*

董韫美
(中国科学院软件研究所计算机科学开放研究实验室, 北京 100080)

摘要 证明了函数类CFRF及其真子类CFPRF分别就是上下文无关语言(CFL)上的偏递归函数和原始递归函数. 讨论了它们与其他论域上定义的递归函数的关系, 指出自然数上的函数和字上函数都是CFL上的函数. 给出了若干常用的字上原始递归函数,包括逻辑连接词和条件式,还给出构造原始递归函数用的强有力算子:受囿极大和受囿极小算子. 构造了两个非平凡的有重要用途的算法. 即任意CFL的特征函数,以及CFL句子的语法分解函数. 基于它们,叙述了扩展和限制函数论域的方法.

关键词 CFRF CFPRF 上下文无关语言的递归函数 受囿极值算子

1 CFPRF和CFRF定义的合宜性

按照文献[1]中的定义,得到了CFL上的两个函数类CFPRF和CFRF. 有什么根据说CFPRF就是CFL上的原始递归函数,而CFRF就是CFL上的递归函数呢?

下文中,当函数的定义域和值域被限定为某具体的CFL时(设为L),就把相应的函数类记为CFPRF_L和CFRF_L. 此外,记全体自然数(包含0在内)组成的集合为$\mathcal{N}$,$\mathcal{N}$上的原始递归函数集合为**P**,$\mathcal{N}$上的递归函数集合为**R**.

采用Gödel-Kleene的定义[2]. 即原始递归函数类**P**是包含3种基本函数:零函数$O(x)=0$,后继函数$S(x)=x+1$,投影函数$U_i^n(x_1,\cdots,x_n)=x_i$,并且封闭于代入算子和原始递归算子的最小函数类;递归函数类**R**是用定义**P**的各种手段和极小算子而得到的. 以下的讨论假定了自然数集合上递归函数论的知识[3].

有时需要在平凡语言上作详细讨论,通常用大写希腊字母Π,Σ表示其终极符号集合. 令非空字母表$\Sigma=\{o_1,\cdots,o_m\}$. Σ上的平凡语言L_Σ由Σ上的所有字组成,或者说它由下面(用BNF表示)的产生式所定义:

$$X ::= \lambda \mid Xo_1 \mid \cdots \mid Xo_m \mid$$

显然,任何以Σ为其终极符号集的CFL是平凡语言L_Σ的子集.

论证CFPRF和CFRF定义合宜性的基本思路如下.

2000-10-23收稿,2001-08-06收修改稿
* 国家自然科学基金资助项目(批准号: 69873042)

首先在 1.1 小节中证明，对于任何平凡语言 L_Σ，存在函数类间的等价对应关系：

$$\mathrm{CFPRF}_{L_\Sigma} \leftrightarrow \mathbf{P},$$

$$\mathrm{CFRF}_{L_\Sigma} \leftrightarrow \mathbf{R}.$$

其次，在 3.3 小节中将证明，对任何 CFPRF(CFRF)函数类中的函数 f，均可以扩展其定义域，使 f 成为在增广字母表 Π 的平凡语言 L_Π 上定义的函数，使得

$$\text{CFPRF 函数 } f \in \mathrm{CFPRF}_{L_\Pi} \quad \text{和} \quad \text{CFRF 函数 } f \in \mathrm{CFRF}_{L_\Pi}$$

注意平凡语言 L_Σ 和 L_Π 不过是 CFL 的特殊情形，故

$$\mathrm{CFPRF}_{L_\Sigma} \text{ 是 CFPRF 函数类} \quad \text{和} \quad \mathrm{CFRF}_{L_\Sigma} \text{ 是 CFRF 函数类}$$

于是

$$\mathbf{P} \leftrightarrow \mathrm{CFPRF}_{L_\Sigma} \text{ 是 CFPRF 函数类；任何 CFPRF 函数} \in \mathrm{CFPRF}_{L_\Pi} \leftrightarrow \mathbf{P}$$

$$\mathbf{R} \leftrightarrow \mathrm{CFRF}_{L_\Sigma} \text{ 是 CFRF 函数类；任何 CFRF 函数} \in \mathrm{CFRF}_{L_\Pi} \leftrightarrow \mathbf{R}.$$

由此导出结论：把 CFPRF 函数类命名为上下文无关语言上的原始递归函数，把 CFRF 函数类命名为上下文无关语言上的递归函数，是完全合理而正当的.

1.1 P(R)与 $\mathrm{CFPRF}_{L_\Sigma}$($\mathrm{CFRF}_{L_\Sigma}$) 间的等价对应

本小节将证明，对任何平凡语言 L_Σ，在 **P** 与 $\mathrm{CFPRF}_{L_\Sigma}$及 **R** 与 CFRF_{L_Σ}之间存在着等价对应关系. 为了对问题给出精确的陈述，先有以下的

引理 1　任何正整数 N，都有惟一的一组正整数系数 $p_n,\cdots,p_0$，满足 $1 \leqslant p_i \leqslant m$，使得

$$N = \sum_{i=0}^{n} p_i \cdot m^i$$

其中 m 是事先给定的任意正整数.

证　略[1].

取 m 为字母表 Σ 中元素个数. 根据引理 1，可以定义映射 $L_\Sigma N: L_\Sigma \to \mathcal{N}$

$$\begin{cases} L_\Sigma N(\lambda) = 0 \\ L_\Sigma N(zo_i) = m \times L_\Sigma N(z) + i \end{cases}$$

定义映射 $NL_\Sigma: \mathcal{N} \to L_\Sigma$

$$\begin{cases} NL_\Sigma(0) = \lambda \\ NL_\Sigma(x+1) = NL_\Sigma(\mathrm{qt}(m,x))\mathrm{letter}(\mathrm{rm}(m,x)+1) \end{cases}$$

其中，$\mathrm{qt}(m,x)$是 x 被 m 除的商的整数部分，$\mathrm{rm}(m,x)$是 x 被 m 除的余数部分，即有

$$x = m \times \mathrm{qt}(m,x) + \mathrm{rm}(m,x).$$

$\mathrm{letter}: \mathcal{N} \to \Sigma$ 是如下映射：$\mathrm{letter}(i) = o_i,\ i = 1,\cdots,m.$

易于证明它们具有如下 4 个性质：

如 $x \in \mathcal{N}$，则 $L_\Sigma N(NL_\Sigma(x)) = x$；如 $z \in L_\Sigma$，则 $NL_\Sigma(L_\Sigma N(z)) = z$；如 $NL_\Sigma(x_1) = NL_\Sigma(x_2)$，则 $x_1 = x_2$；如 $L_\Sigma N(z_1) = L_\Sigma N(z_2)$，则 $z_1 = z_2$.

---

1) 本文中略去的证明均含于中国科学院软件研究所计算机科学实验室技术报告 ISCAS-LCS-2k-03 中

即通过 NL_Σ 和 $L_\Sigma N$,在自然数集合 $\mathcal{N}$ 与平凡语言 L_Σ 之间建立了一一对应关系,且 NL_Σ 和 $L_\Sigma N$ 互为逆映射.

以下将证明,对于 $\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$ 中的任何一个函数 $f: L_\Sigma \times \cdots \times L_\Sigma \to L_\Sigma$,都可以有效地构造出 $\mathbf{P}(\mathbf{R})$ 中的函数 $f^*: \mathcal{N} \times \cdots \times \mathcal{N} \to \mathcal{N}$. 反之,对于 $\mathbf{P}(\mathbf{R})$ 中的任何一个函数 f^*,也都可以有效地构造出 $\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$ 中的函数 f,使得对偶关系

$$\begin{cases} f^*(L_\Sigma N(z_1), \cdots, L_\Sigma N(z_n)) = L_\Sigma N(f(z_1, \cdots, z_n)) \\ f(NL_\Sigma(x_1), \cdots, NL_\Sigma(x_n)) = NL_\Sigma(f^*(x_1, \cdots, x_n)) \end{cases}$$

成立.

于是在 $\mathbf{P}$ 与 $\mathrm{CFPRF}_{L_\Sigma}$ 及 $\mathbf{R}$ 与 CFRF_{L_Σ} 之间,存在着构造性的一一对应关系. 也就是说,$\mathbf{P}(\mathbf{R})$ 与 $\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$ 等价.

下文中,称满足上述对偶关系的一对函数 f 和 f^* 互为对偶函数.

等价性证明思路如下:逐个检查 $\mathbf{P}(\mathbf{R})$,$\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$ 的函数定义手段,构造对偶函数,并且逐个情形验证对偶关系.

首先指出,利用 $L_\Sigma N(NL_\Sigma(x)) = x$, $NL_\Sigma(L_\Sigma N(z)) = z$,可以证明以下 4 个关系式是等价的(互为充分必要条件):

(ⅰ) $f(NL_\Sigma(x_1), \cdots, NL_\Sigma(x_n)) = NL_\Sigma(f^*(x_1, \cdots, x_n))$

(ⅱ) $f^*(L_\Sigma N(z_1), \cdots, L_\Sigma N(z_n)) = L_\Sigma N(f(z_1, \cdots, z_n))$

(ⅲ) $f(z_1, \cdots, z_n) = NL_\Sigma(f^*(L_\Sigma N(z_1), \cdots, L_\Sigma N(z_n)))$

(ⅳ) $f^*(x_1, \cdots, x_n) = L_\Sigma N(f(NL_\Sigma(x_1), \cdots, NL_\Sigma(x_n)))$

于是,检验对偶关系,只需证明 4 个关系式之一成立即可.

以下逐个情形验证对偶关系.

(a) 零函数: $O(x) = 0, x \in \mathcal{N}$

对偶函数: $\mathrm{empty}(z) =_{df} \lambda \quad z \in L_\Sigma$

对偶关系: $\mathrm{empty}(NL_\Sigma(x)) = NL_\Sigma(O(x))$

(b) 后继函数: $S(x) = x + 1, x \in \mathcal{N}$

对偶函数是 L_Σ 中的后继函数 $\sigma(z): L_\Sigma \to L_\Sigma$.

$$\begin{cases} \sigma(\lambda) =_{df} o_1 \\ \sigma(zo_i) =_{df} zo_{i+1} \quad i \neq m \\ \sigma(zo_m) =_{df} \sigma(z)o_1 \end{cases}$$

对偶关系: $S(L_\Sigma N(z)) = L_\Sigma N(\sigma(z))$

今后可以从空字 λ 开始,通过重复应用 σ,枚举出 L_Σ 中所有的字来. 这种枚举方式称为 σ 枚举. 不影响本小节的目的,为行文方便,我们规定:CFRF_{L_Σ} 的极小算子求值采用 σ 枚举方式. 注意,对于 L_Σ,σ 枚举与文献[1]中所述的 CFL 分层枚举稍有不同. 即对于不同长度的两个字,两种枚举都是短者在先,长者在后;对于相同长度的两个字,σ 枚举依正向(即从左向

右看)词典序,CFL 分层枚举依逆向(即从右向左看)词典序来定两者的先后[1].

(c) $P(R)$中的投影函数: $U_i^{n*}(x_1,\cdots,x_n) = x_i,\ x_i\in\mathcal{N},1\leqslant i\leqslant n.$

对偶函数: $U_i^n(z_1,\cdots,z_n) = z_i, z_i\in L_\Sigma, 1\leqslant i\leqslant n.$

对偶关系: $U_i^{n*}(x_1,\cdots,x_n) = L_\Sigma N(U_i^n(NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n)))$.

(d) 代入算子.

设函数 $f^*\in\mathbf{P}(\mathbf{R})$是通过已知函数 $h^*, g_1^*,\cdots,g_r^*$ 代入得到的, 即

$$f^*(x_1,\cdots,x_n) = h^*(g_1^*(x_1,\cdots,x_n),\cdots,g_r^*(x_1,\cdots,x_n))$$

函数 $f\in \mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$是通过已知函数 $h, g_1,\cdots,g_r$ 代入得到的, 即

$$f(z_1,\cdots,z_n) = h(g_1(z_1,\cdots,z_n),\cdots,g_r(z_1,\cdots,z_n),$$

其中,函数 $h^*, g_1^*,\cdots,g_r^*\in\mathbf{P}(\mathbf{R})$分别与函数 $h, g_1,\cdots,g_r\in\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$互为对偶函数.

易知对偶关系成立.

$$f^*(L_\Sigma N(z_1),\cdots,L_\Sigma N(z_n)) = L_\Sigma N(f(z_1,\cdots,z_n))$$

(e) 常字函数.

$\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$ 中的常字函数: $\mathrm{const}_w(z_1,\cdots,z_n) = w, w\in L_\Sigma$

在 $\mathbf{P}(\mathbf{R})$中可以用对后继函数进行多次代入的方法来定义任意常数函数. 于是, $\mathbf{P}(\mathbf{R})$中对偶函数: $\mathrm{const}_w^*(x_1,\cdots,x_n) =_{df} L_\Sigma N(\mathrm{const}_w(NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n)))$

无需再检验对偶关系.

(f) $\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$中的连接函数: $\mathrm{concate}(z_1,\cdots,z_n) = z_1\cdots z_n$

只需讨论两个自变元的简单情形便可, 即: $\mathrm{con}(z_1,z_2) = z_1z_2$

一般情形可利用代入算子实现,即

$$\mathrm{concate}(z_1,\cdots,z_n) = \mathrm{con}(\cdots\ \mathrm{con}(z_1,z_2),\cdots\ z_n)$$

con 定义为[2]

$$\begin{cases}\mathrm{con}(z_1,\lambda) = z_1\\ \mathrm{con}(z_1,\ z_2o_i) = \mathrm{con}(z_1,z_2)o_i\end{cases}$$

$\mathbf{P}(\mathbf{R})$中的函数 con^* 是其对偶函数.

$$\begin{cases}\mathrm{con}^*(x,0) = x\\ \mathrm{con}^*(x,y+1) = m\times\mathrm{con}^*(x,\mathrm{qt}(m,y)) + \mathrm{rm}(m,y) + 1\end{cases}$$

易证对偶关系 $\mathrm{con}^*(L_\Sigma N(z_1),L_\Sigma N(z_2)) = L_\Sigma N(\mathrm{con}(z_1,z_2))$成立.

---

1) 其实, 文献[1]中所述的 CFL 分层枚举, 完全由被枚举语言的文法所决定. 只要使用 L_Σ的另一个定义, 即

$$X ::= \lambda \mid o_1X \mid \cdots \mid o_mX$$

后继函数 $\sigma(z): L_\Sigma\to L_\Sigma$定义为

$$\begin{cases}\sigma(\lambda) =_{df} o_1\\ \sigma(o_iz) =_{df} (\mathrm{last}_m(o_iz)\to\sigma(\mathrm{cutl}(o_iz))o_1,\ \mathrm{cutl}(o_iz)o_{i+1})\\ i = 1,\cdots,m.\end{cases}$$

定义中所用函数和记法见 2.1,2.2, 则两种枚举完全一致. 即不同长度的两个字, 短者在先; 相同长度的两个字, 依正向词典序

2) 利用代入和投影函数, 不难知这里的做法,即在第 2 个自变元位置上使用原始递归, 是允许的

(g) $\mathrm{CFPRF}_{L_\Sigma}$($\mathrm{CFRF}_{L_\Sigma}$)中用联立递归式定义的 l 个函数

$$f_1,\cdots,f_l: L_\Sigma \times \cdots \times L_\Sigma \to L_\Sigma$$

$$\begin{cases} f_i(\lambda, z_2,\cdots,z_n) =_{df} h_i(z_2,\cdots,z_n) \\ f_i(zo_k, z_2,\cdots,z_n) =_{df} h_{i,k}(z, z_2,\cdots,z_n,\cdots,f_j(z, z_2,\cdots,z_n),\cdots) \\ i,j = 1,\cdots,l;\ k = 1,\cdots, m. \end{cases}$$

其中 h_i, $h_{i,k}$是 $\mathrm{CFPRF}_{L_\Sigma}$($\mathrm{CFRF}_{L_\Sigma}$)中的已知函数.

设它们在 $\mathbf{P}(\mathbf{R})$中的对偶函数分别是 h_i^*, $h_{i,k}^*$,则 $f_1,\cdots,f_l$在 $\mathbf{P}(\mathbf{R})$中的对偶函数分别是由以下的联立串值递归式定义的函数:

$$f_1^*,\cdots,f_l^*: \mathcal{N}\times\cdots\times\mathcal{N}\to\mathcal{N}$$

$$\begin{cases} f_i^*(0, x_2,\cdots,x_n) =_{df} h_i^*(x_2,\cdots,x_n) \\ f_i^*(x+1, x_2,\cdots,x_n) =_{df} h_{i,\mathrm{rm}(m,x)+1}^*(\mathrm{qt}(m,x), x_2,\cdots,x_n,\cdots,f_j^*(\mathrm{qt}(m,x), x_2,\cdots,x_n),\cdots) \\ i,j = 1,\cdots,l. \end{cases}$$

根据文献[3],联立串值递归式定义最终可以化归为原始递归式定义.

施归纳于 z, 易于证明对偶关系成立:

$$f_i^*(L_\Sigma N(z), L_\Sigma N(z_2),\cdots,L_\Sigma N(z_n)) = L_\Sigma N(f_i(z, z_2,\cdots,z_n)), i = 1,\cdots,l.$$

(h) $\mathbf{P}(\mathbf{R})$中用原始递归算子定义的函数 f^*:$\mathcal{N}\times\cdots\times\mathcal{N}\to\mathcal{N}$

$$\begin{cases} f^*(0, x_2,\cdots,x_n) = g^*(x_2,\cdots, x_n) \\ f^*(x+1, x_2,\cdots,x_n) = h^*(x, x_2,\cdots,x_n, f^*(x, x_2,\cdots,x_n)) \end{cases}$$

其中, g^*, h^*是 $\mathbf{P}(\mathbf{R})$中已知函数.

设 g^*在 $\mathrm{CFPRF}_{L_\Sigma}$($\mathrm{CFRF}_{L_\Sigma}$)中的对偶函数是 g, 则 f^*在 $\mathrm{CFPRF}_{L_\Sigma}$($\mathrm{CFRF}_{L_\Sigma}$)中的对偶函数 $f: L_\Sigma\times\cdots\times L_\Sigma\to L_\Sigma$可以经由以下方式定义.

首先定义 $\mathbf{P}$ 中函数 H^*:

$$\begin{cases} H^*(0, u, x_2,\cdots,x_n, v) = v \\ H^*(l+1, u, x_2,\cdots,x_n, v) = h^*(l+u, x_2,\cdots,x_n, H^*(l, u, x_2,\cdots,x_n, v)) \end{cases}$$

施归纳于 l, 易证如下性质:

$$H^*(l, \mathrm{qt}(m,x), x_2,\cdots,x_n, f^*(\mathrm{qt}(m,x), x_2,\cdots,x_n)) = f^*(l+\mathrm{qt}(m,x), x_2,\cdots,x_n).$$

此性质的一个直接推论是

$$H^*(x+1-\mathrm{qt}(m,x), \mathrm{qt}(m,x), x_2,\cdots,x_n, f^*(\mathrm{qt}(m,x), x_2,\cdots,x_n))$$
$$= f^*(x+1, x_2,\cdots,x_n).$$

令 $h_k^*(u, x_2,\cdots,x_n, v) = H^*(x+1-\mathrm{qt}(m,x), u, x_2,\cdots,x_n, v)$,当 $\mathrm{rm}(m,x)+1=k$,则易见

$$h_{\mathrm{rm}(m,x)+1}^*(\mathrm{qt}(m,x), x_2,\cdots,x_n, f^*(\mathrm{qt}(m,x), x_2,\cdots,x_n)) = f^*(x+1, x_2,\cdots,x_n).$$

设 h_k^*在 CFPRF_Σ 中的对偶函数是 h_k, $k=1,\cdots,m$,则在$\mathrm{CFPRF}_{L_\Sigma}$($\mathrm{CFRF}_{L_\Sigma}$)中,由以下的原始递归式

$$\begin{cases} f(\lambda, z_2,\cdots,z_n) = g(z_2,\cdots,z_n) \\ f(zo_k, z_2,\cdots,z_n) = h_k(z, z_2,\cdots,z_n, f(z, z_2,\cdots,z_n)) \\ k = 1,\cdots,m \end{cases}$$

定义的函数 f 是 f^* 的对偶函数.

施归纳于 x,很容易证明对偶关系成立:

$$f(NL_\Sigma(x),NL_\Sigma(x_2),\cdots,NL_\Sigma(x_n)) = NL_\Sigma(f^*(x,x_2,\cdots,x_n)).$$

(i) 极小算子.

在 **R** 中用极小算子从已知函数 g^* 定义未知函数 f^*:

$$f^*(x_1,\cdots,x_n) = \mu_y^*[g^*(y,x_1,\cdots,x_n) = 0],\ y,\ x_1,\ \cdots,x_n \in \mathcal{N};$$

即

$$f^*(x_1,\cdots,x_n) = \begin{cases} \text{最小的 } y \text{ 使得} & (\text{ⅰ}) \text{ 对所有 } \tilde{y} \leqslant y, g^*(\tilde{y},x_1,\cdots,x_n) \text{ 有定义,且} \\ & (\text{ⅱ}) \ g^*(y,x_1,\cdots,x_n) = 0, \text{如果这种 } y \text{ 存在,} \\ \text{无定义} & \text{这种 } y \text{ 不存在.} \end{cases}$$

重温文献[1] 中 3.1 小节,在 CFRF_{L_Σ} 中用极小算子由已知函数 g 定义函数 f:

$$f(z_1,\cdots,z_n) =_{df} \mu_z[g(z,z_1,\cdots,z_n)] \quad z,z_1,\cdots,z_n \in L_\Sigma$$

即

$$f(z_1,\cdots,z_n) = \begin{cases} \text{Firstof}\{z \mid g(z,z_1,\cdots,z_n) = \lambda\} \text{ 使得} \\ \qquad (\text{ⅰ}) \text{ 对所有 } \tilde{z} \prec z,\ g(\tilde{z},z_1,\cdots,z_n) \text{ 有定义,且} \\ \qquad (\text{ⅱ}) \ g(z,\ z_1,\cdots,\ z_n) = \lambda, \text{ 如果这种 } z \text{ 存在,} \\ \text{无定义} \quad \text{这种 } z \text{ 不存在.} \end{cases}$$

依前面后继函数及 σ 枚举所述,Firstof$\{z\}$是用后继函数 σ,从 λ 开始逐个枚举 L_Σ 的句子时,集合$\{z\}$中首先被举出的元素. $\prec$ 是 σ 枚举的顺序关系.

$$\text{令 } g(z,z_1,\cdots,z_n) =_{df} NL_\Sigma(g^*(L_\Sigma N(z),L_\Sigma N(z_1),\cdots,L_\Sigma N(z_n)))$$

即 g,g^* 互为对偶函数. 于是

$$g^*(y,x_1,\cdots,x_n) = 0 \text{ 当且仅当 } g(NL_\Sigma(y),NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n)) = \lambda$$

故 $f^*(x_1,\cdots,x_n)$有定义, 当且仅当 $f(NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n))$有定义.

设 $f^*(x_1,\cdots,x_n)$有定义, 且 $f^*(x_1,\cdots,x_n) = y_0$. 当然 $f(NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n))$也有定义, 设 $f(NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n)) = z_0$. 注意, y_0 是使 $g^*(y,x_1,\cdots,x_n) = 0$ 成立的 y 中最小者, 而且

$$g(NL_\Sigma(y_0),\ NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n)) = \lambda$$

z_0 是用后继函数 σ,从 λ 开始,逐个枚举 L_Σ 的句子 z 时,第 1 个被举出并且使 $g(z_0,NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n)) = \lambda$ 成立者.

现在比较 y_0 和 $L_\Sigma N(z_0)$,注意两个后继函数 S 和 σ 互为对偶函数. 再者,对偶关系保证了单调性. 亦即,$z_1 \prec z_2$(z_1 先于 z_2 被 σ 所枚举)的充分必要条件是,$L_\Sigma N(z_1) < L_\Sigma N(z_2)$. 因此,$y_0 = L_\Sigma N(z_0)$. 故

$$f^*(x_1,\cdots,x_n) = y_0 = L_\Sigma N(z_0) = L_\Sigma N(f(NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n)))$$

对偶关系成立.

如 $f^*(x_1,\cdots,x_n)$无定义, 那么 $f(NL_\Sigma(x_1),\cdots,NL_\Sigma(x_n))$也无定义. 极小算子检查完.

至此,我们完成了对 **P**,**R**,$\mathrm{CFPRF}_{L_\Sigma}$,$\mathrm{CFRF}_{L_\Sigma}$ 中定义函数的各个手段的检查, 这就完成了

$$\mathrm{CFPRF}_{L_\Sigma} \leftrightarrow \mathbf{P} \text{ 和 } \mathrm{CFRF}_{L_\Sigma} \leftrightarrow \mathbf{R}$$

的证明.

1.2 平凡语言上递归函数的其他定义手段

证明了 **P** 和 $\mathrm{CFPRF}_{L_\Sigma}$之间存在等价对应关系后,**P** 中定义原始递归函数的手段,都可以移用到 $\mathrm{CFPRF}_{L_\Sigma}$中,仍然有效. 特别是下文将指出,可以使用 σ 原始递归式、串值递归式. 其他手段不难仿照讨论. 我们将在今后径自使用,而对其可用性不加证明.

1.2.1 σ 原始递归式

定义 $f_1,\cdots,f_l: L_\Sigma \times \cdots \times L_\Sigma \to L_\Sigma$的 σ 原始递归式形如:

$$\begin{cases} f_i(\lambda, z_2,\cdots,z_n) =_{df} g_i(z_2,\cdots,z_n) \\ f_i(\sigma(z), z_2,\cdots,z_n) =_{df} h_i(z,z_2,\cdots,z_n,\cdots,f_j(z,z_2,\cdots,z_n),\cdots) \\ i,\ j=1,\cdots,l. \end{cases}$$

其中 g_i, h_i是 $\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$中的已知函数.

设 g_i, h_i在 **P**(**R**)中的对偶函数分别是 g_i^* 和 h_i^*. 按自然数递归函数论的知识,在 **P**(**R**)中,可以使用以下的定义

$$\begin{cases} f_i^*(0,x_2,\cdots,x_n) =_{df} g_i^*(x_2,\cdots,x_n) \\ f_i^*(x+1,x_2,\cdots,x_n) =_{df} h_i^*(x,x_2,\cdots,x_n,\cdots,f_j^*(x,x_2,\cdots,x_n),\cdots) \\ i,j=1,\cdots,l. \end{cases}$$

设 $f_1^*,\cdots,f_l^*$ 在 $\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$中的对偶函数分别是 $F_1,\cdots,F_l$. 则以下对偶关系成立:

$$F_i(z,z_2,\cdots,z_n) = NL_\Sigma(f_i^*(L_\Sigma N(z), L_\Sigma N(z_2),\cdots,L_\Sigma N(z_n)))$$

易知

$$F_i(\lambda,z_2,\cdots,z_n) = g_i(z_2,\cdots,z_n)$$
$$F_i(\sigma(z),z_2,\cdots,z_n) = h_i(z,z_2,\cdots,z_n,\cdots,F_j(z,z_2,\cdots,z_n)),\cdots).$$

恰是定义 f_i 用的 σ 原始递归式.

因此可以认为,前面 σ 原始递归式定义的函数 $f_1,\cdots,f_l$,就是 $f_1^*,\cdots,f_l^*$ 在 $\mathrm{CFPRF}_{L_\Sigma}$($\mathrm{CFRF}_{L_\Sigma}$)中的对偶函数 $F_1,\cdots,F_l$.

今后将不加说明地在 $\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$中使用 σ 原始递归式.

1.2.2 串值递归式

按照自然数递归函数论的知识,在 **P**(**R**)中,可以使用联立串值递归式,即在某自变元值位上的函数值,是利用任意多个或全体在该值位之前的函数值来定义的. 此外,参数允许用已知函数代入.

对应地,在 $\mathrm{CFPRF}_{L_\Sigma}(\mathrm{CFRF}_{L_\Sigma})$中定义函数值时,允许利用按 σ 枚举时某自变元的任意多个(或全体)在该值位之前的函数值[1),并且允许参数用已知函数代入.

假定想要用以下公式来定义函数 $f_1,\cdots,f_l: L_\Sigma \times \cdots \times L_\Sigma \to L_\Sigma$,即

1) 作为特殊情形,这包括了在定义中利用真子字上的函数值,因为每个字的真子字,总是先于该字被 σ 枚举

$$\begin{cases} f_i(\lambda, z_2, \cdots, z_n) = {}_{df} g_i(z_2, \cdots, z_n) \\ f_i(\sigma(z), z_2, \cdots, z_n) = {}_{df} h_i(z, z_2, \cdots, z_n, \cdots, f_j(\phi_r(z), \cdots, \gamma_s(z_t), \cdots, z_n), \cdots) \\ i, j = 1, \cdots, l;\ 2 \leqslant t \leqslant n;\ r, s = 1, 2, \cdots. \end{cases} \tag{1.1}$$

其中 g_i, h_i, ϕ_r 和 γ_s 是 $\mathrm{CFPRF}_{L_\Sigma}$($\mathrm{CFRF}_{L_\Sigma}$)中的已知函数，$\phi_r(z) = z$，或 $\phi_r(z)$ 先于 z 被 σ 所枚举.

$$\begin{cases} f_i(\lambda, z_2, \cdots, z_n) = {}_{df} g_i(z_2, \cdots, z_n) \\ f_i(zo_k, z_2, \cdots, z_n) = {}_{df} h_{i,k}(z, z_2, \cdots, z_n, \cdots, f_j(\phi_r(zo_k), \cdots, \gamma_s(z_t), \cdots, z_n), \cdots) \\ i, j = 1, \cdots, l; k = 1, \cdots, m; 2 \leqslant t \leqslant n; r, s = 1, 2, \cdots. \end{cases} \tag{1.2}$$

其中 g_i, $h_{i,k}$, ϕ_r, γ_s 是 $\mathrm{CFPRF}_{L_\Sigma}$($\mathrm{CFRF}_{L_\Sigma}$)中的已知函数，$\phi_r(z)$ 是 z 的真子字.

以下指出，(1.1)和(1.2)式都是可以使用的，且当 $g_i, h_i, h_{i,k}, \phi_r$ 和 $\gamma_s \in \mathrm{CFPRF}_{L_\Sigma}$ 时，$f_1, \cdots, f_l \in \mathrm{CFPRF}_{L_\Sigma}$. 论证如下.

设 $g_i, h_i, h_{i,k}, \phi_r$ 和 γ_s 在 $\mathbf{P}(\mathbf{R})$ 中的对偶函数分别是 $g_i^*, h_i^*, h_{i,k}^*, \phi_r^*, \gamma_s^*$.

注意前面已经指出，对偶关系保证单调性，即如果 $\phi_r(z)$ 先于 z 被 σ 所枚举，则 $\phi_r^*(x) < x$. 按照自然数递归函数论的知识，在 $\mathbf{P}(\mathbf{R})$ 中，可以使用以下的定义(1.3)和(1.4)

$$\begin{cases} f_i^*(0, x_2, \cdots, x_n) = {}_{df} g_i^*(x_2, \cdots, x_n) \\ f_i^*(x+1, x_2, \cdots, x_n) = {}_{df} h_i^*(x, x_2, \cdots, x_n, \cdots, f_j^*(\phi_r^*(x), \cdots, \gamma_s^*(x_t), \cdots, x_n), \cdots) \\ i, j = 1, \cdots, l; 2 \leqslant t \leqslant n;\ r, s = 1, 2, \cdots. \end{cases} \tag{1.3}$$

$$\begin{cases} f_i^*(0, x_2, \cdots, x_n) = {}_{df} g_i^*(x_2, \cdots, x_n) \\ f_i^*(x+1, x_2, \cdots, x_n) = {}_{df} h_{i,\mathrm{rm}(m,x)+1}^*(\mathrm{qt}(m, x), x_2, \cdots, x_n, \cdots, f_j^*(\phi_r^*(x+1), \\ \qquad \cdots, \gamma_s^*(x_t), \cdots, x_n), \cdots) \\ i, j = 1, \cdots, l;\ k = 1, \cdots, m; 2 \leqslant t \leqslant n;\ r, s = 1, 2, \cdots. \end{cases} \tag{1.4}$$

且当 $g_i^*, h_i^*, h_{i,k}^*, \phi_r^*, \gamma_s^* \in \mathbf{P}$ 时，$f_1^*, \cdots, f_l^* \in \mathbf{P}$.

施归纳于 z，易知：(1.3)式定义的 $\mathbf{P}$ 中函数 $f_1^*, \cdots, f_l^*$，其在 $\mathrm{CFPRF}_{L_\Sigma}$ 中的对偶函数 $f_1, \cdots, f_l$ 满足(1.1)；(1.4)式定义的 $\mathbf{P}$ 中函数 $f_1^*, \cdots, f_l^*$，其在 $\mathrm{CFPRF}_{L_\Sigma}$ 中的对偶函数 $f_1, \cdots, f_l$ 满足(1.2)式.

2　常用函数和算子

对于任何字母表 Σ 上的平凡语言 L_Σ，已经证明了在 $\mathrm{CFPRF}_{L_\Sigma}$ 与 $\mathbf{P}$ 之间和在 CFRF_{L_Σ} 与 $\mathbf{R}$ 之间，存在等价对应. 因此，自然数上的函数和字上函数，都是 CFL 上的函数.

本节给出若干有用的字上函数，包括逻辑连接词和条件式，还给出用来构造函数的算子：受囿极大和受囿极小算子，并依序构造出它们的定义.

2.1　逻辑连接词和条件式

任何有穷集合都是 CFL，很容易定义一个以真、假值为其仅有句子的语言，例如 $\mathrm{BOOLE} = \{\mathbf{t}, \mathbf{f}\}$. 对定义在字母表 $\{o_1, \cdots, o_m\}$ 上的平凡语言 L，可以将 L 中的空串 λ 解释为真值($\mathbf{t}$)，任

何非空的串解释为假值(**f**).

函数 truth: $L\to$BOOLE 将求取 L 句子的真假值,

$\mathrm{truth}(\lambda)=\mathbf{t}$

$\mathrm{truth}(xo_i)=\mathbf{f}\quad i=1,\cdots,m.$

今后我们将在需要时把空串 λ 解释为真值(**t**),把任何非空的串解释为假值(**f**),于是函数也自动具有谓词的地位.

条件函数: $\mathrm{if}(x,y,z)=y$,如果 x 取真值; $\mathrm{if}(x,y,z)=z$,如果 x 取假值.

$\mathrm{if}\ L\times L\times L\to L$

$\mathrm{if}\ (\lambda,y,z)=y$

$\mathrm{if}\ (xo_i,y,z)=z$[1)]

逻辑连接词 $\neg$, $\wedge$, $\vee$, $\supset$, $\equiv$:

$\neg: L\to L$

$\neg(x)=\mathrm{if}(x,o_1,\lambda)$ 此处 o_1 是 Σ 中一指定符号

$\wedge,\vee,\supset,\equiv: L\times L\to L$

$\wedge(x,y)=\mathrm{if}(x,y,x)$

$\vee(x,y)=\mathrm{if}(x,\lambda,y)$

$\supset(x,y)=\mathrm{if}(x,y,\lambda)$

$\equiv(x,y)=\wedge(\supset(x,y),\supset(y,x))$

在前述真假值约定下,直接由定义知下表成立:

x	$\neg(x)$
f	**t**
t	**f**

x	y	$\wedge(x,y)$	$\vee(x,y)$	$\supset(x,y)$	$\equiv(x,y)$
f	**f**	**f**	**f**	**t**	**t**
f	**t**	**f**	**t**	**t**	**f**
t	**f**	**f**	**t**	**f**	**f**
t	**t**	**t**	**t**	**t**	**t**

此表与通常的逻辑连接词表完全相同,因此我们采用了同样的记法.

在不导致混淆时,有时记 $\neg x=_{df}\neg(x)$. 对 $\wedge$, $\vee$, $\supset$, $\equiv$ 也使用中缀记法,如 $x\wedge y=_{df}\wedge(x,y)$,余类推. 易知 $\vee$ 满足结合律,即 $(x\vee y)\vee z=x\vee(y\vee z)$,故今后可写 $x_1\vee\cdots\vee x_n$. 对 $\wedge$ 也同样. 经常也把条件函数 $\mathrm{if}(x,y,z)$ 写成 if x then y else z. 还使用以下的记法,并称之为条件式:

$$(x_1\to y_1,\cdots,x_n\to y_n,z)=_{df}\ \mathrm{if}\ \ x_1\ \ \mathrm{then}\ y_1\ \mathrm{else}$$
$$\vdots$$
$$\mathrm{if}\ \ x_n\ \ \mathrm{then}\ y_n\ \mathrm{else}\ z$$

这函数取 y_{i+1} 为值,当且仅当 $x_1,\cdots,x_i$ 均非 λ , 而 $x_{i+1}=\lambda$;如果 $x_1,\cdots,x_n$ 均非 λ,则结果是 z. 倘若能够保证 $x_1\vee\cdots\vee x_n=\lambda$,则可以写

$$(x_1\to y_1,\ x_2\to y_2,\cdots,x_n\to y_n)$$

此时必取 $y_1,\cdots,y_n$ 之一为值.

1) 此处的 $i=1,\cdots,m$ 被略去不写出. 以下亦同

2.2 若干常用的字上函数

cutf(x)：x 去首一符号

$$\begin{cases}\text{cutf}(\lambda)=\lambda\\ \text{cutf}(o_ix)=x\end{cases}\quad\text{或者}\quad\begin{cases}\text{cutf}(\lambda)=\lambda\\ \text{cutf}(xo_i)=\text{if}\ (x,\lambda,\text{cutf}(x)o_i)\end{cases}$$

cutl(x)：x 去末一符号

$$\begin{cases}\text{cutl}(\lambda)=\lambda\\ \text{cutl}(xo_i)=x\end{cases}\quad\text{或者}\quad\begin{cases}\text{cutl}(\lambda)=\lambda\\ \text{cutl}(o_ix)=\text{if}\ (x,\lambda,o_i\text{cutl}(x))\end{cases}$$

附注　从 cutf 和 cutl 可以看出，同一函数依定义域的不同文法结构，可以有不同的定义方法，在直观性和求值效率上有所区别．对于某些函数求值效率高的文法结构，对于另外的一些函数可能是低效率的；在直观性方面情况也相似．下面给出的函数定义，有时偏重求值效率，有时偏重直观性．

first(x)：x 非 λ 时，为 x 之首一符号

$$\begin{cases}\text{first}(\lambda)=\lambda\\ \text{first}(o_ix)=o_i\end{cases}$$

last(x)：x 非 λ 时，为 x 之末一符号

$$\begin{cases}\text{last}(\lambda)=\lambda\\ \text{last}(xo_i)=o_i\end{cases}$$

$\text{first}_i(x)$：x 之首一符号是 o_i 否？

$$\begin{cases}\text{first}_i(\lambda)=o_1\\ \text{first}_i(o_ix)=\lambda\\ \text{first}_i(o_jx)=o_1\quad\text{当}\quad i\neq j\end{cases}$$

$\text{last}_i(x)$：x 之末一符号是 o_i 否？

$$\begin{cases}\text{last}_i(\lambda)=o_1\\ \text{last}_i(xo_i)=\lambda\\ \text{last}_i(xo_j)=o_1\quad\text{当}\ i\neq j\end{cases}$$

eq(x,y)：$x=y$？

$$\begin{cases}\text{eq}(x,\lambda)=x\\ \text{eq}(x,yo_i)=\text{last}_i(x)\wedge\text{eq}(\text{cutl}(x),y)\end{cases}$$

附注　此处使用了串值递归式来定义，在 1.2.2 中已经指出是被允许的．以后将不再一一指出．

head(x,y)：y 是 x 的端否？

$$\begin{cases}\text{head}(\lambda,y)=y\\ \text{head}(xo_i,y)=\text{head}(x,y)\vee\text{eq}(xo_i,y)\end{cases}$$

tail(x,y)：y 是 x 的尾否？

$$\begin{cases}\text{tail}(x,\lambda)=\lambda\\ \text{tail}(x,yo_i)=\text{if}\ (\text{last}_i(x),\ \text{tail}(\text{cutl}(x),y),o_1)\end{cases}$$

$\sim(x)$：将 x 的符号逆序排列

$$\begin{cases} \sim(\lambda)=\lambda \\ \sim(xo_i)=o_i\sim(x) \end{cases}$$

occur(x,y)：y 在 x 中出现否？

$$\begin{cases} \text{occur}(\lambda,y)=y \\ \text{occur}(xo_i,y)=\text{if}\ (\text{tail}(xo_i,y),\lambda,\text{occur}(x,\ y)) \end{cases}$$

2.2.1 字上函数的性质

以上定义了 11 个字上函数，下面将列出它们的一些基本性质，证明的主要工具是文献[1]中所述联立递归式的结构归纳法.

(ⅰ) 令

$$\begin{cases} f(\lambda)=\lambda \\ f(xo_i)=\text{if}(x,\lambda,f(x)o_i) \end{cases}$$

则有 $f(x)=\text{cutf}(x)$. 这一性质其实是说，函数 cutf 的两个定义是等价的. 同样，函数 cutl 两个定义的等价性即如下的

(ⅱ) 令

$$\begin{cases} f(\lambda)=\lambda \\ f(o_ix)=\text{if}\ (x,\lambda,o_if(x)) \end{cases}$$

则有 $f(x)=\text{cutl}(x)$.

(ⅲ) $x\neq\lambda$ 时，$\text{first}(x)=x$ 之首一符号.

(ⅳ) $x\neq\lambda$ 时，$\text{last}(x)=x$ 之末一符号.

(ⅴ) $\text{first}_i(x)=\lambda$，当且仅当 $\text{first}(x)=o_i$.

(ⅵ) $\text{last}_i(x)=\lambda$，当且仅当 $\text{last}(x)=o_i$.

(ⅶ) $\text{eq}(x,y)=\lambda$，当且仅当 $x=y$.

(ⅷ) $x=\text{cutl}(x)\text{last}(x)$

(ⅸ) $x=\text{first}(x)\text{cutf}(x)$

以下两个性质是说，~ 是字的逆序排列函数.

(ⅹ) $\sim(xy)=\sim(y)\sim(x)$

(ⅹⅰ) $\sim(\sim(x))=x$

(ⅹⅱ) 以下 4 个条件互为充分必要条件：

ⅰ) $\text{head}(x,y)=\lambda$

ⅱ) 有 z 使 $x=yz$ (y 是 x 的端)

ⅲ) 有 $\sim(z)$ 使 $\sim(x)=\sim(z)\sim(y)$

ⅳ) $\text{tail}(\sim(x),\sim(y))=\lambda$

(ⅹⅲ) 以下 4 个条件互为充分必要条件：

ⅰ) $\text{tail}(x,y)=\lambda$

ⅱ) 有 z 使 $x=zy$ (y 是 x 的尾)

ⅲ) 有 $\sim(z)$ 使 $\sim(x)=\sim(y)\sim(z)$

ⅳ) $\text{head}(\sim(x),\sim(y))=\lambda$

利用这两个性质，可以使 head(x,y)和 tail(x,y)互相表达．即

head(x,y) = tail$(\sim(x),\sim(y))$

tail(x,y) = head$(\sim(x),\sim(y))$

（X|V）occur$(x,y)=\lambda$，当且仅当有 u,v 使得 $x=uyv$（y 在 x 中出现，或 y 是 x 的子字）．

（X∨）$x=u_1x_1v_1=u_2x_2v_2$，则 tail$(v_1,v_2)\vee$ tail$(v_2,v_1)=\lambda$，且 head$(u_1,u_2)\vee$ head$(u_2,u_1)=\lambda$．此性质将在 2.3.2 小节中使用，以证明极端子字的存在．

2.3　受囿极大和受囿极小算子

2.3.1　受囿存在量词

给定函数 f，受囿存在量词$(\exists x\leqslant y)[f]$定义为

$$(\exists x\leqslant y)[f(x,x_1,\cdots,x_n)]=_{df}\begin{cases}\lambda & \text{当存在 } y \text{ 的子字 } x,\text{使得 } f(x,x_1,\cdots,x_n)=\lambda\\ \text{非 } \lambda \text{ 字} & \text{否则}\end{cases}$$

函数 $\phi(y,x_1,\cdots,x_n)=(\exists x\leqslant y)[f(x,x_1,\cdots,x_n)]$可以用下式计算：

$$\begin{cases}\phi(\lambda,x_1,\cdots,x_n)=f(\lambda,x_1,\cdots,x_n)\\ \phi(yo_i,x_1,\cdots,x_n)=\phi(\text{cutl}(yo_i),x_1,\cdots,x_n)\vee\phi(\text{cutf}(yo_i),x_1,\cdots,x_n)\vee f(yo_i,x_1,\cdots,x_n)\end{cases}\tag{2.1}$$

引理 2　(2.1)式所定义的函数 ϕ 具如下性质：

$$\phi(y,x_1,\cdots,x_n)=(\exists x\leqslant y)[f(x,x_1,\cdots,x_n)].$$

函数和谓词可以有多种定义方法，例如用受囿存在量词，可以定义

$$\text{occur}(x,y)=(\exists u\leqslant x)(\exists v\leqslant x)[\text{eq}(x,uyv)]$$

$$\text{head}(x,y)=(\exists v\leqslant x)[\text{eq}(x,yv)]$$

$$\text{tail}(x,y)=(\exists u\leqslant x)[\text{eq}(x,uy)]$$

比前面的定义要更加直观．

2.3.2　极端子字

在应用中，常常要找出使受囿存在量词得到满足的 y 的子字．通常结果不是惟一的．我们关心的是其中的某些子字，即

最左极大(LMax)子字：对 y 从左向右看，最先遇到的当中最长的，满足受囿存在量词的 y 的子字．

最右极大(RMax)子字：对 y 从右向左看，最先遇到的当中最长的，满足受囿存在量词的 y 的子字．

最左极小(LMin)子字：对 y 从右向左看，最后遇到的当中最短的，满足受囿存在量词的 y 的子字．

最右极小(RMin)子字：对 y 从左向右看，最后遇到的当中最短的，满足受囿存在量词的 y 的子字．

例　设 $f(x)=\lambda$，当且仅当 x 是由 0 组成的非空串．又设

$$y=a000b00c0000d$$

则 LMax 子字是 a 和 b 间的串 000，RMax 子字是 c 和 d 间的串 0000，LMin 子字是 a 右边第一个 0，RMin 子字是 d 左边第一个 0．

我们将对此概念给出严格的陈述．为此首先指出事实：设有 y 的任意两组分解 u_1,x_1,v_1

和 u_2, x_2, v_2,即 $y = u_1x_1v_1 = u_2x_2v_2$,则由 2.2.1 中最末一个性质,必有

$$(\text{head}(u_1, u_2) \vee \text{head}(u_2, u_1)) \wedge (\text{tail}(v_1, v_2) \vee \text{tail}(v_2, v_1)) = \lambda$$

即 u_1 为 u_2 的端或 u_2 为 u_1 的端,且 v_1 为 v_2 的尾或 v_2 为 v_1 的尾.

对于函数 f 和字 y,可以构造集合 $\text{Decomp}_{f,y}$ 如下:

$$\text{Decomp}_{f,y} = \{(u, x, v) \mid y = uxv \text{ 且} f(x, x_1, \cdots, x_n) = \lambda\},$$

即集合 $\text{Decomp}_{f,y}$ 的元素是 y 关于 f 的受囿分解式,式子中间的子字使受囿存在量词得到满足. 在函数 f 和 y 被确指的情况下,不导致混淆,将用 Decomp 来代替 $\text{Decomp}_{f,y}$,并称其元素为受囿分解式.

$(u, x, v) \in \text{Decomp}$ 称为是(y 关于 f 的)LMax 分解式,其中的 x 称为(y 关于 f 的)LMax 子字,如果满足下面的

LMax 条件:对任何 $(u_1, x_1, v_1) \in \text{Decomp}$,$u$ 是 u_1 的端,且如 $u = u_1$,则 v 是 v_1 的尾.

相似地,还有其他 3 种条件:

RMax 条件. 对任何 $(u_1, x_1, v_1) \in \text{Decomp}$,$v$ 是 v_1 的尾,且如 $v = v_1$,则 u 是 u_1 的端.

LMin 条件. 对任何 $(u_1, x_1, v_1) \in \text{Decomp}$,$v_1$ 是 v 的尾,且如 $v = v_1$,则 u_1 是 u 的端.

RMin 条件. 对任何 $(u_1, x_1, v_1) \in \text{Decomp}$,$u_1$ 是 u 的端,且如 $u = u_1$,则 v_1 是 v 的尾.

这 4 种条件所定义的 4 种子字通称为极端子字.

引理 3 当 Decomp 非空时,每种极端子字均必定存在并且惟一.

特别注意:如 $f(y, x_1, \cdots, x_n) = \lambda$,则 y 的受囿分解式(λ, y, λ)同时满足 LMax 和 RMax 条件;y 如为空字 λ,则其惟一子字 λ 同时满足 4 种极端子字的条件.

2.3.3 受囿极值算子

按照上述极端子字的构造方法,于是有 4 种受囿极值算子.

(ⅰ) 受囿最左极大算子 $\text{LMax}_{x \leqslant y}[f]$

设 $\phi(y, x_1, \cdots, x_n) = \text{LMax}_{x \leqslant y}[f(x, x_1, \cdots, x_n)]$,则

$$\phi(y, x_1, \cdots, x_n) =_{df} \begin{cases} y \text{ 关于 } f \text{ 的 LMax 子字} \underline{x} & (\exists x \leqslant y)[f(x, x_1, \cdots, x_n)] = \lambda \text{ 时} \\ \lambda & y \text{ 的任何子字 } x \text{ 都使} f(x, x_1, \cdots, x_n) \neq \lambda \text{ 时} \end{cases}$$

$\phi(y, x_1, \cdots, x_n)$可以用以下的串值递归式来定义:

$$\begin{cases} \phi(\lambda, x_1, \cdots, x_n) = \lambda \\ \phi(yo_i, x_1, \cdots, x_n) = (f(yo_i, x_1, \cdots, x_n) \to yo_i, \\ \qquad (\exists x \leqslant yo_i)[f(x, x_1, \cdots, x_n) \wedge \text{head}(yo_i, x)] \\ \qquad\quad \to \phi(\text{cutl}(yo_i), x_1, \cdots, x_n), \\ \qquad \phi(\text{cutf}(yo_i), x_1, \cdots, x_n)) \end{cases} \tag{2.2}$$

(ⅱ) 受囿最右极大算子 $\text{RMax}_{x \leqslant y}[f]$

设 $\phi(y, x_1, \cdots, x_n) = \text{RMax}_{x \leqslant y}[f(x, x_1, \cdots, x_n)]$,则

$$\phi(y, x_1, \cdots, x_n) =_{df} \begin{cases} y \text{ 关于 } f \text{ 的 RMax 子字} \underline{x} & (\exists x \leqslant y)[f(x, x_1, \cdots, x_n)] = \lambda \text{ 时} \\ \lambda & y \text{ 的任何子字 } x \text{ 都使} f(x, x_1, \cdots, x_n) \neq \lambda \text{ 时} \end{cases}$$

$\phi(y, x_1, \cdots, x_n)$ 可以用以下的串值递归式来定义:

$$\begin{cases}\phi(\lambda, x_1, \cdots, x_n) = \lambda \\ \phi(yo_i, x_1, \cdots, x_n) = (f(yo_i, x_1, \cdots, x_n) \to yo_i, \\ \qquad (\exists x \leqslant yo_i)[f(x, x_1, \cdots, x_n) \wedge \text{tail}(yo_i, x)] \\ \qquad \to \phi(\text{cutf}(yo_i), x_1, \cdots, x_n), \phi(\text{cutl}(yo_i), x_1, \cdots, x_n))\end{cases} \tag{2.3}$$

（ⅲ）受囿最左极小算子 $\text{LMin}_{x \leqslant y}[f]$

设 $\phi(y, x_1, \cdots, x_n) = \text{LMin}_{x \leqslant y}[f(x, x_1, \cdots, x_n)]$，则

$$\phi(y, x_1, \cdots, x_n) =_{df} \begin{cases} y \text{ 关于 } f \text{ 的 LMin 子字 } \underline{x} & (\exists x \leqslant y)[f(x, x_1, \cdots, x_n)] = \lambda \text{ 时} \\ \lambda & y \text{ 的任何子字 } x \text{ 都使 } f(x, x_1, \cdots, x_n) \neq \lambda \text{ 时} \end{cases}$$

$\phi(y, x_1, \cdots, x_n)$ 可以用以下的串值递归式来定义：

$$\begin{cases}\phi(\lambda, x_1, \cdots, x_n) = \lambda \\ \phi(yo_i, x_1, \cdots, x_n) = ((\exists x \leqslant \text{cutl}(yo_i))[f(x, x_1, \cdots, x_n)] \to \phi(\text{cutl}(yo_i), x_1, \cdots, x_n), \\ \qquad (\exists x \leqslant \text{cutf}(yo_i))[f(x, x_1, \cdots, x_n)] \to \phi(\text{cutf}(yo_i), x_1, \cdots, x_n), \\ \qquad f(yo_i, x_1, \cdots, x_n) \to yo_i, \lambda)\end{cases} \tag{2.4}$$

（ⅳ）受囿最右极小算子 $\text{RMin}_{x \leqslant y}[f]$

设 $\phi(y, x_1, \cdots, x_n) = \text{RMin}_{x \leqslant y}[f(x, x_1, \cdots, x_n)]$，则

$$\phi(y, x_1, \cdots, x_n) =_{df} \begin{cases} y \text{ 关于 } f \text{ 的 RMin 子字 } \underline{x} & (\exists x \leqslant y)[f(x, x_1, \cdots, x_n)] = \lambda \text{ 时} \\ \lambda & y \text{ 的任何子字 } x \text{ 都使 } f(x, x_1, \cdots, x_n) \neq \lambda \text{ 时} \end{cases}$$

$\phi(y, x_1, \cdots, x_n)$可以用以下的串递归式来定义：

$$\begin{cases}\phi(\lambda, x_1, \cdots, x_n) = \lambda \\ \phi(yo_i, x_1, \cdots, x_n) = ((\exists x \leqslant \text{cutf}(yo_i))[f(x, x_1, \cdots, x_n)] \to \phi(\text{cutf}(yo_i), x_1, \cdots, x_n), \\ \qquad (\exists x \leqslant \text{cutl}(yo_i))[f(x, x_1, \cdots, x_n)] \to \phi(\text{cutl}(yo_i), x_1, \cdots, x_n), \\ \qquad f(yo_i, x_1, \cdots, x_n) \to yo_i, \lambda)\end{cases} \tag{2.5}$$

今指出上述串值递归式定义的正确性，仅针对 LMax，其余情形留给读者.

引理 4　(2.2)式所定义的函数 ϕ 具如下性质

$$\phi(y, x_1, \cdots, x_n) = \begin{cases} \lambda & \text{当 } y \text{ 的任何子字 } x \text{ 都使得 } f(x, x_1, \cdots, x_n) \neq \lambda \text{ 时} \\ y \text{ 关于 } f \text{ 的 LMax 子字 } \underline{x} & (\exists x \leqslant y)[f(x, x_1, \cdots, x_n)] = \lambda \text{ 时} \end{cases}$$

定理 1　CFPRF_{L_Σ}封闭于 4 种受囿极值算子.

LMax，RMax，LMin 和 RMin 是非常强有力的原始递归算子，具有很直观的含义. 研究如何在计算机上得到它们的直接而高效率的实现，是很有意义的问题. 目的是得到 CFPRF 的高效率求值算法.

类似地，$\neg$，$\wedge$，$\vee$，$\supset$，$\equiv$，eq，head，tail，occur，if，条件式和受囿存在量词$(\exists x \leqslant y)[f]$都可以直接在计算机上高效实现，这样一方面得到强有力的函数定义手段，另一方面得到高效求值手段.

3 CFPRF 的非平凡例子

有了受囿极大算子之后,可以着手构造一些非平凡的有重要用途的算法. 首先要构造的是任意 CFL 的特征函数,基于它再构造任意 CFL 句子的语法分解函数. 它们都是 CFPRF 类中的函数. 我们需要这些函数来完成函数类等价性证明. 注意,我们只讨论无二义文法.

3.1 CFL 的特征函数

设有文法 $G=(V_N,V_T,X,\mathscr{P})$,它定义的语言是 L. 并不丧失一般性,仅为避免无效计算,可以假设:由 G 不会产生任何形如 $Y\Rightarrow\cdots\Rightarrow Y$ $(Y\in V_N)$的推导.

设字母表 $V_T\subseteq\Sigma$, 把函数 $\mathrm{ch}_L: L_\Sigma\to L_\Sigma$ 称为语言 L 的特征函数, 如果它满足

$$\mathrm{ch}_L(x)=\begin{cases}\lambda & \text{当 } x\in L\\ \text{非 } \lambda \text{ 字} & \text{否则}\end{cases}$$

即 $\mathrm{ch}_L(x)$取真值,当且仅当 x 为 L 中的句子.

事实上,特征函数是成组联立定义的,对 V_N 中每一非终极符都得到一个特征函数,即以该非终极符为开始符号的文法所定义语言的特征函数. 构造方法如下:

对任一非终极符 $Y\in V_N$,设 $\mathscr{P}$中以 Y 为左部的产生式的集合是$\{Y\to P_1,\cdots,Y\to P_n\}$,便可写出特征函数定义式

$$\mathrm{ch}_Y(x)=_{df}\ \mathrm{term}_{P_1}(x)\vee\cdots\vee\mathrm{term}_{P_n}(x)\quad x\in L_\Sigma,$$

其中,每个产生式 $Y\to P_i$ 对应一个 $\mathrm{term}_{P_i}(x)$, $i=1,\cdots,n$, 并具有如下形式:

对于平凡产生式的情形, 即当 P_i 中不含非终极符时,

$$\mathrm{term}_{P_i}(x)=_{df}\mathrm{eq}(x,P_i)\quad x\in L_\Sigma$$

对于非平凡产生式的情形, 即当 $P_i=u_{i1}Y_{i1}u_{i2}Y_{i2}\cdots u_{ir_i}Y_{ir_i}v_i$ 时,其中 v_i 和各 u_{ij}是终极符串,各 Y_{ij}是非终极符,则

$$\begin{aligned}\mathrm{term}_{P_i}(x)=_{df}\ &(\exists x_1\leqslant x)(\exists x_2\leqslant x)\cdots(\exists x_{r_i}\leqslant x)[\mathrm{eq}(x,u_{i1}x_1u_{i2}x_2\cdots u_{ir_i}x_{r_i}v_i)\\ &\wedge\mathrm{ch}_{Y_{i1}}(x_1)\wedge\cdots\wedge\mathrm{ch}_{Y_{ir_i}}(x_{r_i})]\quad x\in L_\Sigma\end{aligned}$$

如上法对 V_N 中所有非终极符 Y 写出 ch_Y 的定义式,它们定义了 $\mathrm{CFPRF}_{L_\Sigma}$中的一组函数:

$$\begin{cases}\mathrm{ch}_Y(x) & Y\in V_N, x\in L_\Sigma\\ \mathrm{term}_{P_i}(x) & P_i \text{ 是 } \mathscr{P}\text{中各产生式的右部}\end{cases}$$

其中的 ch_X 即所要的函数 ch_L, X 是 G 的开始符号. 注意,由 $\mathrm{ch}_Y(x)$和 $\mathrm{term}_{P_i}(x)$的定义知道:

(ⅰ) $\mathrm{ch}_Y(x)=\lambda$, 当且仅当,有一个产生式 $Y\to P_i$, 使得 $\mathrm{term}_{P_i}(x)=\lambda$

(ⅱ) $\mathrm{term}_{P_i}(x)=\lambda$,当且仅当ⅰ)或者ⅱ).

ⅰ) P_i 中不含非终极符,并且 $x=P_i$

ⅱ) P_i 中含非终极符, $P_i=u_{i1}Y_{i1}u_{i2}Y_{i2}\cdots u_{ir_i}Y_{ir_i}v_i$

x 可按 P_i 分解,使得

$$x=u_{i1}x_1u_{i2}x_2\cdots u_{ir_i}x_{r_i}v_i$$

并且

$$\mathrm{ch}_{Y_{ij}}(x_j) = \lambda, j = 1, \cdots, r_i.$$

事实上，有以下的

定理 2　对于所有函数 $\mathrm{ch}_Y(x), \mathrm{term}_{P_i}(x), Y \in V_N$，$P_i$ 是 $\mathscr{P}$ 中各产生式的右部，$x \in L_\Sigma$. 有

（ⅰ）$\mathrm{ch}_Y(x) = \lambda$ 的充分必要条件是：x 是以 Y 为开始符号的文法 $G_Y = (V_N, V_T, Y, \mathscr{P})$ 所定义语言 $L(G_Y)$ 中的句子.

（ⅱ）$\mathrm{term}_{P_i}(x) = \lambda$ 的充分必要条件是：x 可以由产生式 $Y \to P_i$ 推出，即 $Y \to P_i \overset{*}{\Rightarrow} x$.

3.1.1　基于特征函数的 CFL 句子枚举

对于任意非空字母表 Σ，已有枚举 L_Σ 中字的手段，即 1.1 中定义的函数 σ. 对于 L_Σ 所包含的任意上下文无关语言 L，由前述又有了它的特征函数 ch_L. 于是，便有了枚举 L 中句子的另一手段，即定义

$$\begin{cases} \sigma^0(z) =_{df} z & z \in L_\Sigma \\ \sigma^{r+1}(z) =_{df} \sigma(\sigma^r(z)) \end{cases}$$

则 $L_\Sigma = \{\sigma^r(\lambda) \mid r = 0, 1, \cdots\}$. 把 L_Σ 中的句子 $\sigma^r(\lambda)$，按 $r = 0, 1, 2, \cdots$ 之序，逐一交由 L 的特征函数 ch_L 检查. 当该句子是 L 的合法句子时，便予举出；不是，则予抛弃. 这样就枚举了 L 中所有合法句子. 我们称这种枚举方法为 σ-ch_L 枚举.

3.2　CFL 句子的语法分解函数

另一个有用而非平凡的函数例子是语法分解函数.

设有 $V_N, V_T, \mathscr{P}$，设字母表 $\Sigma = V_T \cup \{\lhd, \rhd\} \cup \{\underline{Y} \mid Y \in V_N\}$，我们将在平凡语言 L_Σ 上，定义一组原始递归函数 $\{\mathrm{parse}_Y \mid Y \in V_N\}$. 函数 parse_Y 将对以 Y 为开始符号的文法 $(V_N, V_T, Y, \mathscr{P})$ 所产生语言中的每一个句子，生成相应的推导树.

所谓句子 x 关于文法 G 的推导树，是一个符号串，即在 x 中插入了若干特殊的括号对 $\lhd$ 和 $\rhd \underline{Y}$（可以嵌套）. 括号中的符号串可看成子树，$\rhd$ 右边的 $\underline{Y}$ 用来指出该子树是由非终极符 Y 所推出的.

推导树的例. 为简洁起见，使用 BNF 表示的文法

stmt ∷ = exp

exp ∷ = term | exp + term | exp − term

term ∷ = pri | term * pri | term/pri

pri ∷ = $a \mid b \mid \cdots \mid z$

则句子 $a + b$ 关于此文法（以 stmt 为开始符号）的推导树是

$$\lhd\lhd\lhd\lhd\lhd\ a\ \rhd \underline{\mathrm{pri}} \rhd \underline{\mathrm{term}} \rhd \underline{\mathrm{exp}} + \lhd\lhd\ b\ \rhd\ \underline{\mathrm{pri}} \rhd \underline{\mathrm{term}} \rhd \underline{\mathrm{exp}} \rhd \underline{\mathrm{stmt}}$$

以下叙述定义函数 parse_Y 的具体做法.

对任一非终极符 $Y \in V_N$，设 $\mathscr{P}$ 中以 Y 为左部的产生式的集合是 $\{Y \to P_1, \cdots, Y \to P_n\}$. 任取 $Y \to P_i$.

定义　字 z 可按产生式 $Y \to P_i$ 分解，是指

（ⅰ）平凡产生式的情形，即 P_i 中不含非终极符. 如果 $z = P_i$，此等式称为 z 关于 P_i 的分

解式.

(ⅱ) 非平凡产生式的情形,即 P_i 中含非终极符. 如果

$$P_i = u_{i1}Y_{i1}u_{i2}Y_{i2}\cdots u_{ir_i}Y_{ir_i}v_i$$

$$z = u_{i1}x_1u_{i2}x_2\cdots u_{ir_i}x_{r_i}v_i \qquad Y_{ij}\overset{*}{\Rightarrow}x_j, j=1,\cdots,r_i.$$

其中 v_i 和各 u_{ij}, x_j是终极符串,各 Y_{ij}是非终极符,则等式 $z = u_{i1}x_1u_{i2}x_2\cdots u_{ir_i}x_{r_i}v_i$称为 z 关于 P_i 的分解式,x_j称为 z 关于 P_i 的分解式的 Y_{ij}分量.

根据 CFL 知识, 对于无二义文法, $Y\to P_i\overset{*}{\Rightarrow}z$, 当且仅当 z 关于 P_i 的分解式存在并且惟一.

对非平凡产生式 $Y\to P_i$, $P_i = u_{i1}Y_{i1}u_{i2}Y_{i2}\cdots u_{ir_i}Y_{ir_i}v_i$, 写出定义:

$$\begin{cases}\mathrm{Qcomponent}_{ij}(z,x) =_{df} (\exists x_1\leqslant z)\cdots(\exists x_{j-1}\leqslant z)(\exists x_{j+1}\leqslant z)\cdots(\exists x_{r_i}\leqslant z)\\ \qquad [\mathrm{eq}(z, u_{i1}x_1\cdots u_{ij-1}x_{j-1}u_{ij}x\ u_{ij+1}x_{j+1}\cdots u_{ir_i}x_{r_i}v_i)\\ \qquad \wedge \mathrm{ch}_{Y_{i1}}(x_1)\wedge\cdots\wedge\mathrm{ch}_{Y_{ij-1}}(x_{j-1})\wedge\mathrm{ch}_{Y_{ij}}(x)\\ \qquad \wedge \mathrm{ch}_{Y_{ij+1}}(x_{j+1})\wedge\cdots\wedge\mathrm{ch}_{Y_{ir_i}}(x_{r_i})]\\ \mathrm{component}_{ij}(z) =_{df} \mathrm{LMax}_{x\leqslant z}[\mathrm{Qcomponent}_{ij}(z,x)]\\ \qquad j=1,\cdots,r_i.\end{cases}$$

其中 $\mathrm{ch}_{Y_{ij}}$是特征函数.

定理 3 (ⅰ) $\mathrm{Qcomponent}_{ij}(z,x)=\lambda$ 的充分必要条件是,存在 z 关于 P_i 的分解式,x 是该分解式的 Y_{ij}分量.

(ⅱ) $\mathrm{term}_{P_i}(z)=\lambda$, 则 $\mathrm{component}_{ij}(z)$是 z 关于 P_i 的分解式的 Y_{ij}分量.

今后将称 $\mathrm{component}_{ij}$为分量函数.

现在不妨假设,以 V_N 中非终极符 Y 为左部的产生式集合 $\{Y\to P_1,\cdots,Y\to P_n\}$ 中,前面 s 个为平凡产生式,即 $P_1,\cdots,P_s$ 中不含非终极符;后面 $n-s$ 个为非平凡产生式,即对于 $i=s+1,\cdots,n$, P_i 中含非终极符.

对非终极符 Y,可写出如下的语法分解函数 parse_Y的定义:

$$\begin{aligned}\mathrm{parse}_Y(z) =_{df} &(\mathrm{eq}(z,P_1)\vee\cdots\vee\mathrm{eq}(z,P_s)\to\lhd z\rhd\underline{Y},\\ &\vdots\\ &\mathrm{term}_{P_i}(z)\to\lhd u_{i1}\mathrm{parse}_{Y_{i1}}(\mathrm{component}_{i1}(z))\cdots u_{ir_i}\mathrm{parse}_{Y_{ir_i}}(\mathrm{component}_{ir_i}(z))v_i\rhd\underline{Y},\\ &\vdots\qquad\qquad i=s+1,\cdots,n\\ &\lambda).\end{aligned}$$

对所有非终极符 $Y\in V_N$,得到一组联立串值递归式. 它们定义了一组原始递归函数 $\mathrm{parse}_Y\in\mathrm{CFPRF}_{L_\Sigma}$,并具有如下性质:

设上下文无关文法 $G=(V_N,V_T,\mathscr{P})$定义的语言是 L. 则 $\mathrm{parse}_Y: L_\Sigma\to L_\Sigma$

$$\mathrm{parse}_Y(x) = \begin{cases}x\text{ 关于文法 }G\text{ 的推导树} & \text{当 }x\text{ 是 }L\text{ 的句子时}\\ \lambda & \text{当 }x\text{ 不是 }L\text{ 的句子时}\end{cases}$$

即所得到的函数 parse_Y 能生成 L 中任何句子的推导树.

注意我们做了多少准备工作，才得到函数 parse_Y 的定义.

3.2.1 直接定义的语法分解函数

事实上，可以用一个 CFL 来表示文法 G 的推导树. 该 CFL 的文法不妨叫做 G_{tree}，定义如下：

设文法 $G=(V_N,V_T,Y,\mathscr{P})$，以字母表 $\Sigma=V_T\cup\{\lhd,\rhd\}\cup\{\underline{Y}\mid Y\in V_N\}$ 作为 G_{tree} 的终极符集合，非终极符集合仍旧是 V_N，开始符号是 Y.

如果 $X\to P$ 是 G 的产生式，则 $X\to\lhd P\rhd\underline{X}$ 是 G_{tree} 的产生式.

不难看出，语言 $L(G)$ 的句子的推导树是语言 $L(G_{\text{tree}})$ 的句子.

下面的 CFPRF 函数 $\text{parse}_Y: L(G)\to L(G_{\text{tree}})$ 对 $L(G)$ 的句子求推导树.

对全体 $Y\in V_N$ 列出联立递归式，即对每个以 Y 为左部的产生式 $Y\to P_i$，写出定义式.

P_i 中不含非终极符时，写出

$$\text{parse}_Y(P_i)=\lhd P_i\rhd\underline{Y}$$

$P_i=u_{i1}Y_{i1}\cdots u_{ir_i}Y_{ir_i}v_i$，$Y_{i1},\cdots,Y_{ir_i}$ 是非终极符，$u_{i1},\cdots,u_{ir_i},v_i$ 是终极符串，写出

$$\text{parse}_Y(u_{i1}Y_{i1}\cdots u_{ir_i}Y_{ir_i}v_i)=\lhd u_{i1}\text{parse}_{Y_{i1}}(Y_{i1})\cdots u_{ir_i}\text{parse}_{Y_{ir_i}}(Y_{ir_i})v_i\rhd\underline{Y}$$

这就得到了函数 parse_Y 的定义. 回顾前面，在 L_Σ 上定义特征函数和语法分解函数时，何等费力. 这就是文献[1]中提到的："用一般的字上递归函数不能直接地反映出被加工对象的结构，而在加工过程中又不可避免地要涉及到结构. 为此必须把一个语法分解算法嵌入函数的定义之中，相当累赘."

CFRF 的本质优越性在于，函数定义域和值域的语法结构，可以在函数定义中随意操作和应用. 本小节的例子，证实了这一结论.

3.3 函数论域的扩展

任何 CFPRF(CFRF)函数类中的函数 f，利用前面的特征函数和分量函数，均可以扩展其定义域，使 f 成为在增广字母表 Π 的平凡语言 L_Π 上定义的函数.

确切地说，设有函数 $f: L_1\times\cdots\times L_n\to L$，各语言 $L_1,\cdots,L_n,L$ 的终极符集合分别为 $V_{T_1},\cdots,V_{T_n},V_T$. 设它们均不含元素 $\perp$.

令字母表 $\Pi=V_{T_1}\cup\cdots\cup V_{T_n}\cup V_T\cup\{\perp\}$，$L_\Pi$ 是 Π 上的平凡语言[1]. 要定义 $f^*: L_\Pi\times\cdots\times L_\Pi\to L_\Pi$，使得当 $f\in\text{CFPRF}$ 时，$f^*\in\text{CFPRF}_{L_\Pi}$；$f\in\text{CFRF}$ 时，$f^*\in\text{CFRF}_{L_\Pi}$.

$$f^*(x_1,\cdots,x_n)=\begin{cases}f(x_1,\cdots,x_n) & \text{当 } x_i\in L_i,(i=1,\cdots,n),f(x_1,\cdots,x_n)\text{有定义}\\ \perp & \text{当有某 } i,x_i\notin L_i\\ \text{无定义} & \text{当 } x_i\in L_i,(i=1,\cdots,n),f(x_1,\cdots,x_n)\text{无定义}\end{cases}$$

那么我们就可以认为：CFPRF 函数 $f\in\text{CFPRF}_{L_\Pi}$ 和 CFRF 函数 $f\in\text{CFRF}_{L_\Pi}$.

以下将叙述如何构造性地定义 f^*，只需逐个检查 CFPRF，CFRF 的函数定义手段.

---

1) 为使 CFRF_{L_Π} 中极小算子求值采用的枚举与分层构造枚举一致，关于 L_Π 的文法及其上后继函数 σ 的定义，应按第 257 页脚注 1)所述办法处理

(ⅰ) f 是常字函数、投影函数、连接函数:注意 $f(x_1,\cdots,x_n)\in L\subset L_{\Pi}$. 定义

$f^*(x_1,\cdots,x_n)=_{df}(\mathrm{ch}_{L_1}(x_1)\wedge\cdots\wedge\mathrm{ch}_{L_n}(x_n)\to f(x_1,\cdots,x_n),\perp)\quad x_i\in L_{\Pi},i=1,\cdots,n,$

其中 ch_{L_i} 是语言 L_i 的特征函数.

(ⅱ) 代入算子: 即 $f(x_1,\cdots,x_n)=h(g_1(x_1,\cdots,x_n),\cdots,g_m(x_1,\cdots,x_n))$

其中

$$g_i:L_1\times\cdots\times L_n\to L^{(i)},i=1,\cdots,m$$
$$h:L^{(1)}\times\cdots\times L^{(m)}\to L$$
$$f:L_1\times\cdots\times L_n\to L$$

则 $f^*(x_1,\cdots,x_n)=_{df}h^*(g_1^*(x_1,\cdots,x_n),\cdots,g_m^*(x_1,\cdots,x_n))\quad x_i\in L_{\Pi},i=1,\cdots,n$, 其中, $h^*,g_1^*,\cdots,g_m^*$ 分别是从 $h,g_1,\cdots,g_m$ 经论域扩展所得的函数.

(ⅲ) 联立递归式:即定义了 m 个函数 $f_1,\cdots,f_m:L_1\times\cdots\times L_n\to L$, L_1 由文法 $G_1=(V_N,V_T,X_1,\mathscr{P}_1)$ 产生. 按定义,即对每个产生式 $X_1\to P_i$,有 m 个已知函数 $h_{P_{ij}},j=1,\cdots,m$, 使得当 P_i 中不含非终极符时,有规则组

$$\begin{cases}f_j(P_i,y_2,\cdots,y_n)=h_{P_{ij}}(y_2,\cdots,y_n)\\ j=1,\cdots,m.\end{cases}\qquad(*)$$

当 P_i 含非终极符, 即 $P_i=u_{i1}Y_{i1}u_{i2}Y_{i2}\cdots u_{ir_i}Y_{ir_i}v_i$,其中 $u_{i1},\cdots,u_{ir_i},v_i$ 为终极符串, $Y_{i1},\cdots,Y_{ir_i}$ 为非终极符. 则有规则组

$$\begin{cases}f_j(P_i,y_2,\cdots,y_n)=h_{P_{ij}}(Y_{i1},\cdots,Y_{ir_i},y_2,\cdots,y_n,\cdots,f_k(Y_{il},y_2,\cdots,y_n),\cdots)\\ j,k=1,\cdots,m;1\leqslant l\leqslant r_i.\end{cases}\qquad(**)$$

此时 m 个函数 $f_j^*:L_{\Pi}\times\cdots\times L_{\Pi}\to L_{\Pi}$ 用以下的联立串值递归式定义.

首先不妨假设,以 V_N 中非终极符 X_1 为左部的产生式集合 $\{X_1\to P_1,\cdots,X_1\to P_n\}$ 中, 前面 s 个为平凡产生式. 即 $P_1,\cdots,P_s$ 中不含非终极符;后面 $n-s$ 个为非平凡产生式,即对于 $i=s+1,\cdots,n,P_i$ 中含非终极符. 便可写出如下的定义式组:

$$\begin{cases}f_j^*(y,y_2,\cdots,y_n)=_{df}(\mathrm{eq}(y,P_1)\to h_{P_{1j}}^*(y_2,\cdots,y_n),\cdots,\mathrm{eq}(y,P_s)\to h_{P_{sj}}^*(y_2,\cdots,y_n),\cdots,\\ \qquad\mathrm{term}_{P_i}(y)\to h_{P_{ij}}^*(\mathrm{component}_{i1}(y),\cdots,\mathrm{component}_{ir_i}(y),\\ \qquad\qquad y_2,\cdots,y_n,\cdots,f_k^*(\mathrm{component}_{il}(y),y_2,\cdots,y_n),\cdots),\\ \qquad\qquad\vdots\quad k=1,\cdots,m;i=s+1,\cdots,n;1\leqslant l\leqslant r_i.\\ \qquad,\\ \qquad\perp)\\ j=1,\cdots,m.\end{cases}$$

其中 $h_{P_{ij}}^*$ 是从 $h_{P_{ij}}$ 经论域扩展所得的函数.

(ⅳ) 极小算子:$f(x_1,\cdots,x_n)=\mu_y[g(y,x_1,\cdots,x_n)]$,则 $f^*(x_1,\cdots,x_n)=_{df}\mu_y[g^*(y,x_1,\cdots,x_n)]$, 其中 g^* 是从 g 经论域扩展所得的函数.

3.4 函数论域的限定

有时想把某个函数 $f:L_{\Sigma_1}\times\cdots\times L_{\Sigma_n}\to L_{\Sigma}$ 限制在论域 $L_1\times\cdots\times L_n\to L$ 上研究. 即 n 个自

变元的定义域限为 $L_1 \times \cdots \times L_n$，函数的值域限为 L. 其中上下文无关语言 $L_i \subset L_{\Sigma_i}$ $(i = 1, \cdots, n)$；$L \subset L_\Sigma$；Σ 中不含元素 $\perp$.

可以使用以下的办法来定义. $f^* : L_{\Sigma_1} \times \cdots \times L_{\Sigma_n} \to L_\Sigma \cup \{\perp\}$，注意 $L_\Sigma \cup \{\perp\}$ 是 CFL.

$$f^*(x_1, \cdots, x_n) =_{df} (\mathrm{ch}_L(f(x_1, \cdots, x_n)) \wedge \mathrm{ch}_{L_1}(x_1) \wedge \cdots \wedge \mathrm{ch}_{L_n}(x_n) \to f(x_1, \cdots, x_n), \perp)$$

其中 $\mathrm{ch}_L, \mathrm{ch}_{L_1}, \cdots, \mathrm{ch}_{L_n}$ 是特征函数.

当 $f \in$ CFPRF 时，$f^* \in$ CFPRF；$f \in$ CFRF 时，$f^* \in$ CFRF. 即

$$f^*(x_1, \cdots, x_n) = \begin{cases} f(x_1, \cdots, x_n) & \text{当 } f(x_1, \cdots, x_n) \text{有定义，且 } x_1, \cdots, x_n, f(x_1, \cdots, x_n) \\ & \text{在论域 } L_1 \times \cdots \times L_n \to L \text{ 中} \\ \perp & f(x_1, \cdots, x_n) \text{ 有定义，但 } x_1, \cdots, x_n, f(x_1, \cdots, x_n) \\ & \text{不在论域 } L_1 \times \cdots \times L_n \to L \text{ 中} \\ \text{无定义} & f(x_1, \cdots, x_n) \text{无定义} \end{cases}$$

故函数 f^* 可视为(不是真正的)被定义在 $L_1 \times \cdots \times L_n$ 上. 也就是说，函数 f^* 是通过用论域 $L_1 \times \cdots \times L_n \to L \cup \{\perp\}$ 来约束原来函数 f 而得到的. 当 f 的自变元和函数值都在限定的语言中时，f^* 与 f 相同；否则，f^* 以 $\perp$ 为值，可以理解为没有定义.

于是，在平凡语言上定义的函数、算子，在上述意义下，均可挪用于 CFRF. 但是，函数 f^* 的定义域和值域归根结底仍旧是

$$f^* : L_{\Sigma_1} \times \cdots \times L_{\Sigma_n} \to L_\Sigma \cup \{\perp\}$$

应该特别注意的是，其自变元和函数值分别具有 $L_{\Sigma_1}, \cdots, L_{\Sigma_n}$，$L_\Sigma \cup \{\perp\}$ 中的语法结构，而不具有 $L_1, \cdots, L_n, L$ 中的语法结构.

参　考　文　献

1　董韫美. 上下文无关语言上的递归函数——Ⅰ. CFPRF 及 CFRF 的定义. 中国科学，E 辑，2002，32(1)：103～115

2　Cutland N. Computability: An Introduction to Recursive Function Theory. Cambridge: Cambridge University Press, 1980

3　Péter R. Recursive Functions. New York: Academic Press, 1967